TIVMI V2.1 USER MANUAL

Bruno Dutailly and Pierre Guyomarc'h
Université Bordeaux 1, CNRS, UMR 5199 PACEA, F-33405, Talence Cedex, France
b.dutailly@pacea.u-bordeaux1.fr, pierreguyo@gmail.com

November 19, 2012

Contents

I Introduction **5**

II Graphical Interface **7**

1 The menu bar **7**
1.1 File . . . 7
1.2 Plugin . . . 7
1.3 Script . . . 7
1.4 Options . . . 8

2 3D layout **8**
2.1 3D view . . . 8
2.2 tools . . . 8
2.2.1 Main behavior . . . 8
2.2.2 Groups . . . 9

3 Objects control **9**
3.1 Camera and environment . . . 9
3.2 Image 3D . . . 10
3.3 Point 3D . . . 10
3.4 Segment 3D . . . 11
3.5 Path 3D . . . 11
3.6 Basis 3D . . . 11
3.7 Mesh 3D . . . 12
3.8 Plane . . . 12
3.9 Straight line . . . 12
3.10 Bounding box . . . 12
3.11 Plot . . . 12
3.12 Data . . . 12

III Options 13

IV Plugins 15

4 Creating a 3D Image 15
4.1 Automatic threshold . . . 15
4.2 Simple threshold . . . 15
4.3 Double threshold . . . 16
4.4 Crop a 3D image . . . 16
4.5 Crop a 3D image with a bounding sphere . . . 18
4.6 Erase out of a bounding sphere . . . 18
4.7 Gaussian blur . . . 18
4.8 Ortho slices . . . 20
4.9 Merge . . . 22
4.10 Reslice . . . 23
4.11 Voxellize a mesh . . . 25
4.12 Load Analyze Img and Hdr . . . 25
4.13 Load DICOM . . . 26
4.14 Load PGM3D . . . 27
4.15 Load PNG serie . . . 27
4.16 Load RAW . . . 28
4.17 Load VFF . . . 28

5 Creating a Mesh 29
5.1 Load pts . . . 29
5.2 Load obj . . . 30
5.3 Load stl . . . 30
5.4 Load ply . . . 30
5.5 HMH Surface 2 . . . 31
5.6 Morph mesh . . . 32

6 Creating a 3D Point 32
6.1 Intersection plane and straight line . . . 32
6.2 Orthogonal projection: point on plane . . . 33
6.3 Orthogonal projection: point on straight line . . . 33
6.4 Make a 3D point on a 3D segment from a ratio . . . 33
6.5 Barycenter path 3D . . . 34
6.6 Shift a point along the normal of a plane . . . 34
6.7 Skeleton using region growing . . . 34
6.8 Skeleton using 2D region growing on all slices . . . 35
6.9 Convert path3D to points . . . 35
6.10 Convert segment to 2 points . . . 36
6.11 Point3Ds HMH Segment3D HMH value . . . 36
6.12 Intersection between straight line and mesh . . . 36

7 Creating a 3D path **37**
7.1 Load obj spline . . . 37
7.2 Orthoprojection Path3D on plane . . . 37
7.3 Resample Path3D . . . 37
7.4 Intersection mesh plane . . . 38

8 Creating a Plane **39**
8.1 Align on mesh . . . 39
8.2 Make plane . . . 41
8.3 Medium plane . . . 41
8.4 Orthogonal plane from 2 planes through 1 point . . . 42
8.5 Orthogonal to a plane including a straight line . . . 43
8.6 Orthogonal segment 3D . . . 43
8.7 Make an orthogonal point through 2 points . . . 43
8.8 Parallel to straight line through 2 points . . . 44
8.9 Shift plane to a 3D point . . . 44
8.10 Bench vice a mesh . . . 44
8.11 Constrained Bench vice a mesh . . . 46
8.12 Bench vice a limited mesh . . . 46

9 Creating a Segment 3D **48**
9.1 Make 3D segment from 2 points . . . 48
9.2 Maximum distance from a 3D path to a plane . . . 48
9.3 Minimum distance between 2 path3Ds . . . 48
9.4 Shift along plane normal with ratio . . . 49
9.5 Segment angle spread . . . 49
9.6 Segment ladder . . . 50

10 Creating Data **50**
10.1 Angle 2 planes . . . 50
10.2 Angle 3 points . . . 52
10.3 Height of triangle . . . 52
10.4 Perimeter of a shape in a binary slice . . . 53
10.5 Volume from voxels . . . 54
10.6 Weight of all materials in a 3D image . . . 54

11 Creating Straight Lines **55**
11.1 Make straight line from 2 planes . . . 55
11.2 Make straight line from 1 segment . . . 55
11.3 Shift a straight line to a 3D point . . . 56

12 Output to file **56**
12.1 Reslice and save to radio tiff a 3D image . . . 56
12.2 Project each point of a mesh into an image . . . 56
12.3 Save Path 3D to Svg . . . 57
12.4 Save Image3D to analyze img and hdr files . . . 58

12.5 Save Image3D to amira mesh file format . . . 58
12.6 Save segmented Image3D to amira mesh file format . . . 59
12.7 Save mesh to obj . . . 59
12.8 Save path 3d to obj . . . 59
12.9 Save mesh to ply . . . 60
12.10Save Image 3D to one radio tiff file . . . 60
12.11Save Image 3D to tiff . . . 60
12.12Save an object to tso . . . 61
12.13Save Image 3D to VFF . . . 61

13 Creating any object **61**
13.1 Load tso . . . 61

V Segmentation **62**

14 Creating a segmentation **62**
14.1 From scratch . . . 62
14.2 From a 8 bits 3D image . . . 63

15 Editing a segmentation **63**

16 The segmentation GUI **63**
16.1 Slice view . . . 64
16.2 Materials list . . . 66
16.3 Tools . . . 71
16.3.1 No tool . . . 71
16.3.2 Brush . . . 71
16.3.3 Magic wand . . . 71
16.3.4 Selection manipulation . . . 74
16.3.5 Lasso . . . 76

17 What can I do with a segmentation? **81**

VI Scripts **83**

18 Scripting language **83**
18.1 Commands on objects . . . 83
18.1.1 Image 3D . . . 83
18.1.2 Point 3D . . . 84
18.1.3 Plane . . . 84
18.1.4 Mesh . . . 85
18.1.5 Bounding Box . . . 85
18.1.6 Segment 3D . . . 86
18.1.7 Straight line . . . 86
18.1.8 Path 3D . . . 86

18.1.9 AFA3D . . . 87
18.1.10 Group . . . 88
18.2 Commands to create objects . . . 89
18.3 Regular expressions . . . 90
18.4 Delete objects . . . 90
18.5 Show objects . . . 90
18.6 Hide objects . . . 90
18.7 Rename objects . . . 90
18.8 Call a plug-in . . . 90

19 Script generation **91**

VII Scenarios **92**

20 Open a DICOM exam, and place 3D points on slices **92**

21 Create a segment and find the HMH interface between two tissues **93**
21.1 Create a new segment . . . 93
21.2 Create a segment from 2 existing 3D points . . . 94
21.3 Evaluate the density profile on a segment . . . 94
21.4 Automatically compute the interface 3D points on a 3D segment with the HMH protocol 95

22 Reconstruct two surfaces from two different densities **96**
22.1 Select the region of interest . . . 96
22.2 Reconstruct HMH surfaces . . . 98

23 Enhanced craniometrics **100**
23.1 Plane construction and landmark positioning . . . 100
23.2 Measurement in projection . . . 103
23.3 Measurement of soft tissue depth . . . 104

24 Segmentation of a 3D image: basic threshold **106**

25 Advanced segmentation of a 3D image **108**

Part I
Introduction

TIVMI, standing for Treatment and Inscreased Vision for Medical Imaging, is a software developed in the PACEA laboratory, UMR 5199 of CNRS, Université Bordeaux 1, and Ministère de la Culture. It is designed for the need of anthropology researchers for accurate and reproducible measurements in 3D of medical imaging material, when other software does not respond to this goal. It also has been extended to mesh and point cloud manipulation, and has met

needs of archaeologists.
This help is designed to beginners and advanced users. Paragraphs in **bold** are simple descriptions and explainations, the rest being more detailed.

Part II
Graphical Interface

1 The menu bar

1.1 File

to load or save states. It is an internal file format of TIVMI. You can save all objects and re-open them. It creates a .tsf file: tsf is for TIVMI Script File, and a directory with the same name (without .tsf). All objects are stored in the directory, and the tsf file contains instructions to open objects files. If you want to share data with another computer, copy the tsf and the corresponding folder with the same name.
Objects in the folder are saved as tso files. tso is for Tivmi Serialized Object. It consists in a binary format of the object. In the future, state file format will evolve to a gzip file containing xml and binary files to make interoperability easier with others softwares.

1.2 Plugin

contains features that allow to create new objects from other objects. You will find a description of all plug-ins on the website. Plug-ins in this menu are sorted by the object type they produce. e.g. if you want to create a plane, go to the plane submenu, and select the desired operation to create a plane. e.g. if you want to open DICOM files, that means you want to create a 3D Image, etc. To export objects, use the "output to file" submenu.
Plugins are dll files for Windows, and .so files for linux. They are stored in the plugins directory, and loaded dynamically when TIVMI starts. Live update can replace or add plug-ins without modifying the TIVMI program.

1.3 Script

let you record and replay scripts (.tsf) files. The created file is plain text, so you can easily edit those.
"Generate and execute" is useful to repeat the same commands several times.

- Iterate: will repeat the command written in the text area, and replace all "%d" occurrences by the number between "from" and "to". e.g. if a plug-in generates 10 3D points, and you need to show them, type: "`Show p%d`" and fill from=1 and to=10. This will generate 10 lines: SHOW p1, SHOW p2, ... Show p10, and execute them.

- regular expression: to learn what is a regular expression, we invite you to read the wikipedia article about it: http://en.wikipedia.org/wiki/Regular_expression

In the text area, type the command, and add, inside the command, a regular expression between two "#". All objects names matching the regular expression will be played in the command. e.g. to show all objects which a name beginning by "p" and followed by a number, type: "`Show #^p[0-9]+$#`" It means: show all object beginning (`^`) by a p followed by a digit (from 0 to 9), at least one time (+), with nothing else ($).

See 18 and 19 for more details about available commands, syntax, and generation.

1.4 Options

is a basic graphical interface to read and edit the TIVMI configuration file. Useful if you need to change default folders for states, or setup proxy informations for live update if tou are behind a proxy (see III).

2 3D layout

2.1 3D view

- left mouse button and movement = rotation
- middle mouse button and movement = pan (translation parallel to the point of view)
- right mouse button and movement up and down (or forward and backward) = zoom in and out
- TIP: if you want to rotate the object in a clockwise way from the point of view, use the left mouse button, and draw small counter-clockwise circles
- Hold "Ctrl" key and click on an object to center the view on the targeted point
- see 3.1 for more customization of 3D view

2.2 tools

Tools are a tree containing all objects in the TIVMI memory. They are sorted by type. The first column is the name of the object and next to the name, you will find actions.

2.2.1 Main behavior

1. Name of the object: a double click on it will bring up the object control window and select the correct object in the list.
2. Eye: show or hide the object.

3. Hard drive: swap the object. It writes the object to the hard drive to save memory. You can "unswap" it by clicking again on it
4. Red cross: delete the object.
5. Pencil: rename the object.
6. Target: center the camera point of view on the object. This is equivalent to go to the "camera and environment" window control, select the object, and "set target" (see 3.1).

You can select more than one object using a multiple selection (shift and/or control + click) and click on an icon to apply the action to all selected objects.

2.2.2 Groups

You can group several objects in a group, even if objects are not of the same type. This is useful to collapse a group of objects and save space in the tree, and also to hide or show all objects without a tedious multiple selection.
Right click on an object or on a multiple selection, and click on "add to group" → "New group". Give a name and click on OK.
You can also add objects to an existing group through the same menu.
A right click on an object inside a group will offer you to remove the object from the group.
Very few plugins use groups, but this will evolve later.

3 Objects control

Each sub window contains on the top a combo list of available objects.
All controls will apply to the selected object chosen on this list. If the list is empty, it means that there is no object of this type OR the object is hidden! Show it (see 2.2) and it will appear.
I will describe only some features, I let you discover others.
You also must know that EVERY object must be named with a unique name.

3.1 Camera and environment

- set target centers the camera on the selected object. Useful to rotate around easily, or simply recover it in the viewing space. e.g. a DICOM file contains a 3D location and TIVMI respects this information. So the DICOM slices may appear far from the origin (0,0,0).
- FOV: field of view. It is the horizontal angle of the camera.
- Near and far: the near and far planes are orthogonal planes to the camera. In front of the near plane, and behind the far plane, objects are not drawn. You can use this to virtualy cut objects (increase near plane). Note that the 3D picking accuracy is linked to the distance between near and far

plane. So be careful: if you increase the far plane too much, the 3D picking will not be accurate.

- Stereo: the 3D view will generate an image for each eye. You must have a NVidia Quadro display card or any display card compatible with opengl quad buffered stereo.
- Environment: controls lighting and background color.

3.2 Image 3D

Is a stack of classical 2D images. Often, it corresponds to a medical scanner exam.

- Slice scroll bar lets you navigate in slices.
- Info gives you the image dimensions, and some DICOM informations if available.
- Resolution allows you to read and modify the voxel resolution. As given in DICOM files, TIVMI distinguishes Z voxel resolution and space between slices, but for all treatments, only space between slices is taken into account.
- windowing controls the contrast and brightness with a "window on data". Center and width are equivalent to the double horizontal slider under the image histogram. Usefull for 16 bits images.
- You can show or hide the image bounding box.
- Segmentation: You can use the button to convert a 8 bits 3D image to a segmentation when needed and start the segmentation GUI (see V).

3.3 Point 3D

Is a point in 3D space with coordinates x, y, and z. The point can be displayed in different shapes, sizes and colors.

- Z-Slice Style is a special shape for 3D points picked on 3D images where image stack is oriented on Z. It consists in 2 squares on each side of a slice, connected by a sgement crossing the slice exactly at the point coordinates. The ball style is more conventionnal.
- Coordinates: let you convert world coordinates of a 3D point to local coordinates of a 3D image where Z is the slice number, ans X and Y are coordinates in the 2D image.
  Coordinates and size are writable and can be validated by the button apply.

- 3D picking: chose an object that will intercept the click on the object reference combo list. Click on "pick new" (or shortcut key "p"), and click on the object in the 3D view. You will be prompted for a name. The 3D point will be automatically shown (you don't need to click on show/hide). Use pick move each time you want to move the selected point. Note that hitting space will reiterate the last action performed (this can be useful when the point needs to be moved several times, or if numerous point must be places in a row, keeping your eyes on the 3D view.

- "copy to clipboard": copy ALL 3D points in the clipboard, and not only the 3D point selected. Then you can "paste" in Excel to get all coordinates. Be carefull, numbers are written with a floating point (.). On french Excel, you will have to replace "." by "," to be able to manipulate those entries as numbers.

- "export": copy the selected 3D points in a textfile with a specified separator. We suggest you to use the easier "copy to clipboard" button.

3.4 Segment 3D

is an association of two 3D points connected. TIVMI gives the Euclidian distance between the 2 points, which is the length of the segment. It can be displayed as different sizes and colors. A color gradient from one point to the other gives the order between points.
3D picking works like 3D point but you need to click 2 points. The name will be asked between the 2 clicks in the 3D view. You can rotate the object and "pick move point 2" if you need to pick a non visible position.

3.5 Path 3D

Is a path of 3D points connected like a chain of segments. An option can close the path.
The "lock action" checkbox allows you to click more than one time in the 3D view to pick new points. Check this box, and click on Insert after, then click click click.... in the 3D view. When finished, or if you want to rotate or navigate, uncheck the box, change the view, recheck the box, and re-enter pick new. This can be used to collect outline data (explore the Path submenu in plug-ins for further treatments).

3.6 Basis 3D

Is a 3D orthonormal basis. It is used to compute basis changes on 3D point coordinates.
Reserved to advanced use of some plugins.

3.7 Mesh 3D

Is a polygonal mesh composed of points and faces. TIVMI is able to compute normals per face or normals per vertex, and invert them. It is also possible to scale the mesh and to optimize the vertex use (delete duplicates). Color, opacity, and shininess are also customizable. TIVMI is able to manage point clouds, i.e. meshes without faces.
You can rebuild normals if the illumination is bad. Normal per vertex will smooth visually (no degradation of data). Normal per face will show you the reality of the mesh, but it may slows the 3D view, and the rendering of the mesh will not look pretty.

3.8 Plane

Is a 3D plane using the equation ax+by+cz+d=0. It can be displayed as different colors and thickness. Its normal can be displayed.
Even if the plane is shown, you need to constraint it to a specific object because a 3D plane is by definition infinite, and TIVMI needs to know where to draw it. So choose a target and apply the constraint. You can set a thickness in the normal way. It applies in real time. This is usefull to simulate a parallel plane without creating it.

3.9 Straight line

Is an infinite line. TIVMI uses a system of 2 planes equations to store a straight line. It corresponds to the intersection of 2 planes. It can be displayed as different colors.
Use as plane (see 3.8).

3.10 Bounding box

Bounding box is usefull to crop a 3D image in order to save memory (e.g. before 3D reconstruction). To create it, click on new, and click 2 opposite corners in the 3D view on the reference object.

3.11 Plot

Is a 2D plot serie using 2 axis. The plot can be displayed as an histogram or a curve, and mathematical functions can be applied to enhance the curve (LOG, square root, etc).
Used for segment profiles or 3D Image histograms.

3.12 Data

Data: contains numerical data produced by some plugins (e.g. volume computation).

Part III
Options

In TIVMI, options are saved into the "tivmi.ini" file located in your home directory in a folder named ".TIVMI". On Windows and linux platform, folders starting with a dot are hidden. If you want to browse these folders, you must setup your file browser to let you navigate inside.

TIVMI has a little graphical interface to setup the configuration file without editing directly the "tivmi.ini" file. In the menu bar, click on "option" and then "edit".

- folders

  Contains a set of 4 folders that TIVMI uses to store the information about different objects and actions:

  - Exports: when using the export button in the Point 3D of Segment 3D object controls (see 3.3 and 3.4).
  - Scripts: when loading or recording scripts (see 1.3).
  - States: when loading or saving states (see 1.1).
  - Swap: when swapping objects (see 2.2).

- license

  This contains a reference to the network card needed for the license checking. You shouldn't need to modify this.

- live

  Contains informations for the live assistant. The live assistant will replace the live updater and will provide a graphical interface to manage the license. You shouldn't need to modify this.

- mdiDisplayConfiguration

  This contains the configuration of mdi windows. Mdi windows are the window system of object control. You shouldn't need to modify this. The only option you can edit is the "docked" one. Set it to "true" to dock the mdi window inside TIVMI. Else, set it to "false". This is useful if you use dual screens. In this case, the object control window should be on the second screen. If you go back to a single screen configuration, the object control window is lost. Set the docked option to false, close TIVMI, and restart it.

- updater

  Contains informations for the live update. This will be replaced by the live section.

  - proxy let you configure the updater to use a proxy connection: first, setup the enable switch to “true”. Then fill “host”, “password”, “port”, and “username”.
  - autoUpdate: this switch enable (set to “true”) or disable (set to false) the updater at each TIVMI startup.
  - Directory: do not modify
  - Server: do not modify

- AllowedThreadCount is the number of thread that TIVMI is allowed to use on multithread processings. Yous should set it to the core count of your system minus 1 to let you use your system during a computation. You can find the number of available cores on your system in the task manager.

- BackgroundColor: you should not modify this here but in the camera and environment window (see 3.1).

Part IV
Plugins

Each plugin has input(s) and output(s). Generally, outputs are names of new objects to create (or prefix for multiple results). It can be also a filename for the "output to file" plugins. Inputs are existing objects in TIVMI, or numerical values. It means that an input object can be a hidden object. When you run a plugin, you will always have the same window:

- on the left, input objects. You must chose the adapted object in a combo list. For numerical values, you will find a text input to fill, and a combo list. The combolist contains numerical values in TIVMI memory. You can choose a numerical value here, but if you fill the input text, the combobox value will be ignored.
- on the right, output names to fill with unique names, or filename to browse.

Keep in mind that plugins will never alterate input objects. They always create new objects and don't modify existing objects.

4 Creating a 3D Image

4.1 Automatic threshold

- Input:
  1. 3D image
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_AutomaticThreshold_Image3D`

Computes the mean between the minimum and maximum grey levels of the input image, and thresholds the image. Under or equal to the mean, grey level 0 is stored, and over the mean, a lavel of 255 is stored. The new image is 8 bits depth and keeps the position, orientation, and voxel size of the input 3D image.

4.2 Simple threshold

- Input:
  1. 3D image
  2. Grey value
- Output:

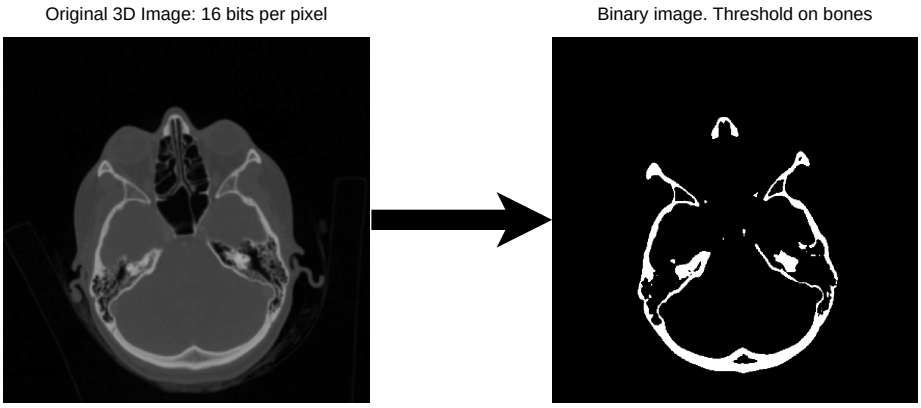

Figure 1: Thresholded 3D image (only one slice shown)

1. Name of new 3D image

- Callable name in scripts: `Image3D_SimpleThreshold_Image3D`

For less than the grey value, grey level 0 is stored, more or equal to the value, the level 255 is stored (see figure 1). The new image is 8 bits depth and keeps the position, orientation, and voxel size of the input 3D image.

4.3 Double threshold

- Input:
  1. 3D image
  2. Grey value 1
  3. Grey value 2
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_DoubleThreshold_Image3D`

Less than the value 1 and more than the value 2, grey level 0 is stored, and between or equal to the two values, 255 is stored (see figure 2). The new image is 8 bits depth and keeps the position, orientation, and voxel size of the input 3D image.

4.4 Crop a 3D image

- Input:
  1. 3D image

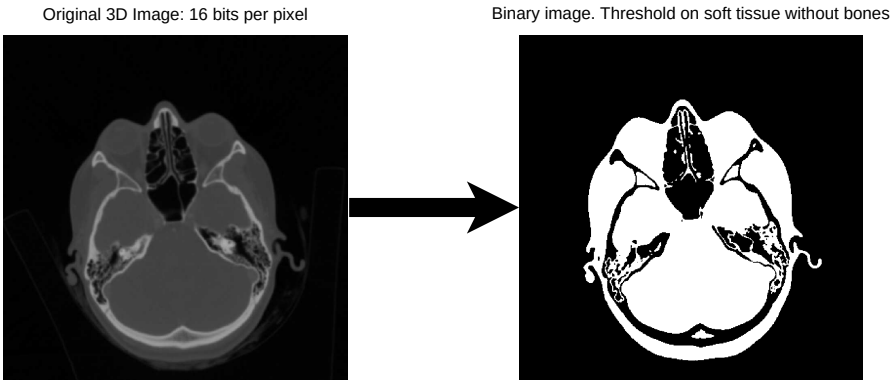

Figure 2: Double thresholded 3D image (only one slice shown)

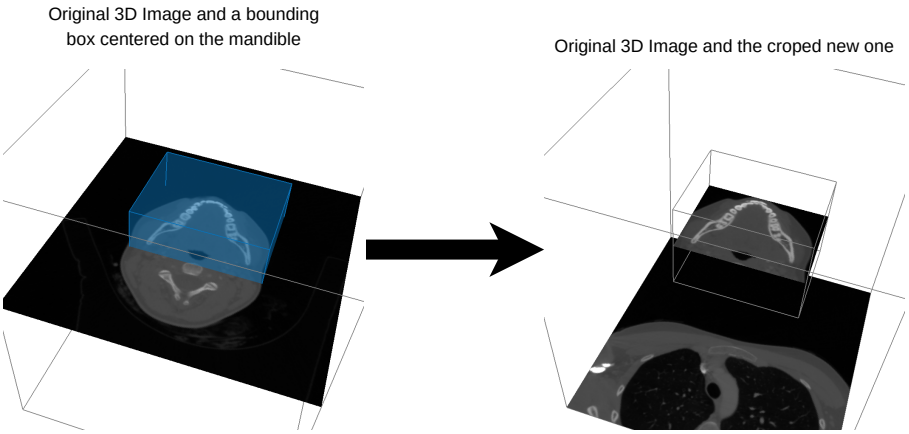

Figure 3: Crop a 3D image using a bounding box

2. Bounding Box

- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_Crop_Image3D`

Crop the image and only keep the voxels inside the bouding box (see figure 3). If a voxel is cut by the bounding box, it is kept. If the new image has odd dimensions (width, height or slice count), a column, a line or slice is added to make a 3D image with pair dimensions. The new 3D image is located and oriented as the original one.

4.5 Crop a 3D image with a bounding sphere

- Input:
  1. 3D image
  2. Bounding sphere radius
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_CropBoundingSphere_Image3D`

The sphere is decribed by the second input: a 3D segment. The first point of the segment is the sphere center, and the second point is on the sphere surface. The new image has the dimension of the smallest bounding box containing the sphere. If the new image has odd dimensions (width, height or slice count), a column, a line or slice is added to make a 3D image with pair dimensions. The new 3D image is located and oriented as the original one.
This plugin can be combined with the "erase out of a bounding sphere" plugin (see 4.6).

4.6 Erase out of a bounding sphere

- Input:
  1. 3D image
  2. Bounding sphere radius
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_EraseOutOfBoundingSphere_Image3D`

Sets to 0 all voxels which geometric center is outside the bouding sphere (see fig 4).
This plugin is useful to limit a study to a sphere volume, like volume computation, tissues analysis, etc.

4.7 Gaussian blur

- Input:
  1. 3D image
  2. Teta
- Output:
  1. Name of new 3D image

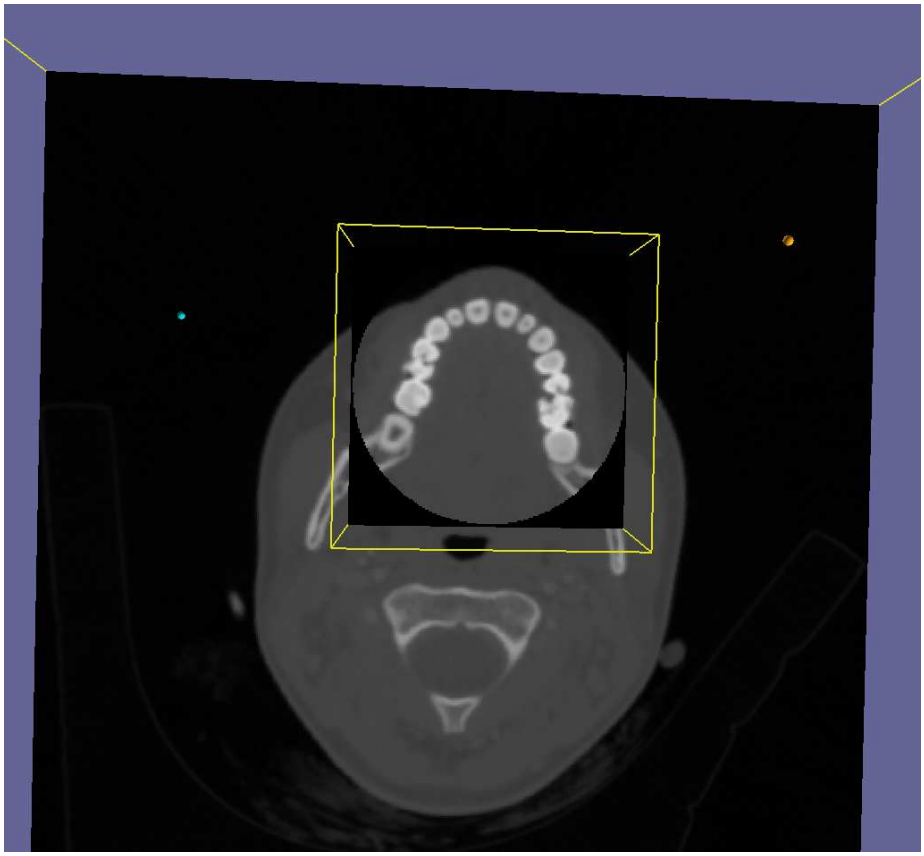

Figure 4: Voxels out of the sphere are set to 0

- Callable name in scripts: `Image3D_GaussianBlur_Image3D`

Compute a 3D mask of $6 * round(theta) + 1$ size. Each cell of the mask contains a value computed using a gaussian function.
Considering that mask dimension is always odd and the center of the mask has coordinates: ($c_x = 0$, $c_y = 0$, $c_z = 0$), the value stored in a cell (x,y,z) has the value:
$\frac{1}{\pi.2.theta^2} * e^{\frac{-(x^2+y^2+z^2)}{2.theta^2}}$
This plugin is not equivalent to a 2D gaussian blur on each slice. It is a real 3D gaussian blur.
Then each voxel of the new image is the original voxel convolved by the mask scaled to a ratio of 1.
The new image has the same depth, position, orientation, and voxel size of the input 3D image.
This plugin can be used to reduce noise on images.
Complexity is exponential. This plugin is multi-threaded, but it still can take a while even on a powerfull computer.

4.8 Ortho slices

- Input:
  1. 3D image
- Output:
  1. Name of first new 3D image
  2. Name of second new 3D image
- Callable name in scripts: `Image3D_OrthoSlice_Image3D`

Compute 2 new images orthogonally to the original one (see figure 6).

The first image is oriented with the X axis unchanged. It means that in the new image, the X axis will be the same as the original. So the new Y axis will correspond to the old Z, and the new Z will correspond to the old Y.
The second image is oriented with the Y axis unchanged. It means that in the new image, X axis will correspond to the old Z, the Y is unchanged, and the Z will correspond to the old X.
If the original image is an axial view, and the lateral axis of the specimen is the X axis in the image, the the first orthoslice will be a frontal view, and the second will be a sagittal view. Otherwise, if you make a mistake, you can rename any object in TIVMI (see 2.2).
Created images are correctly located and oriented. The most important is to note that in the case of non-isometric voxels, new images have rectangular pixels (see figure 7). An isometric voxel has the same length, width and height. In most medical data set, it is not the case. DICOM images have square pixels,

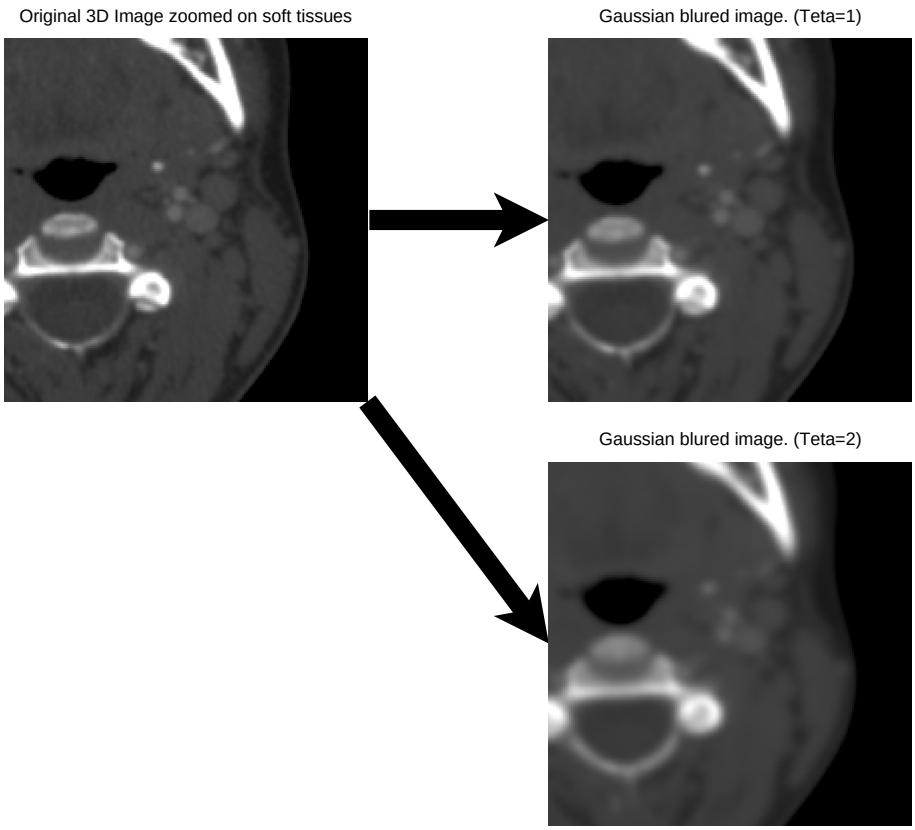

Figure 5: Gaussian blur on a 3D image with 2 different values of theta

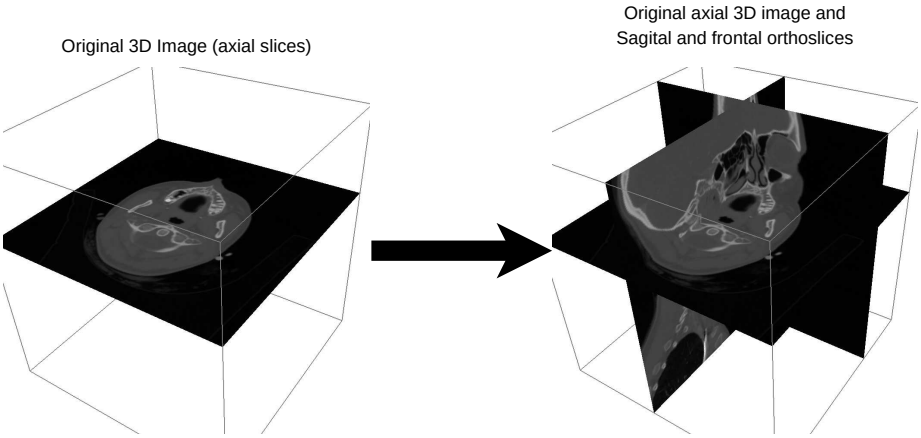

Figure 6: Orthoslices

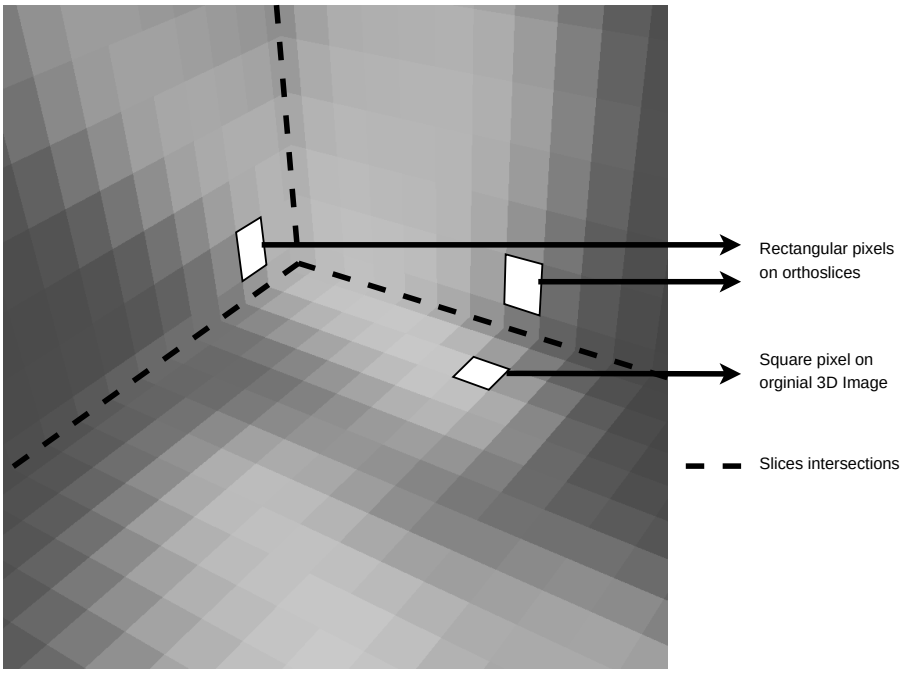

Figure 7: Zommed view of orthoslices on non isometrics voxels.

but the distance between slices is different, and often greater than the pixel size. An orthoslice image will have rectangular pixels. In TIVMI, orthoslice images are stored as squared pixels, but displayed as rectangular pixels. The more you zoom on an orthoslice, the more you will see those rectangular pixels. Computations on orthoslices are perfectly equivalent to the same computations on the original data set. Be careful with the storage: classical file formats for images like jpeg, png, tiff, etc support only square pixels. TIVMI will compute a new image resampled to store in those file formats. The pixel size will be the smallest side of the rectangular pixel. In the internal file format of TIVMI (tso file format), TIVMI keeps the information of rectangular pixels, and the image will not be resampled.

4.9 Merge

- Input:
  1. Image 3D reference
  2. Image 3D to add
  3. Use 3D positionning
- Output:

1. Name of new 3D image

- Callable name in scripts: `Image3D_Merge_Image3D`

Merge two 3D images into one. If use 3D positionning is set at 0, the new 3D image dimensions are herited from the ref image and the slice count is the sum of slices of both images. If images cross themselves, slices are intercalated. If slices of the image to add are shifted compared to the the reference image, the shift will be kept.
If the option is set to 1, the plugin will look at the exact position of slices in 3D space and compute a bounding box containig both images. This bounding box defines the new image dimensions. If the image to add is shifted and its location in the 3D space is correct, the new computed image should not reproduce the shift.
This plugin is usefull if you need to join two different DICOM sets of a unique object splited in different series.

4.10 Reslice

- Input:
  1. Image 3D to reslice
  2. Use basis
  3. 3D Basis
  4. 3D Plane of cuts
  5. Number of cut
  6. Space between cuts
  7. Pixel subdivision
  8. Pixel size
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_Reslice_Image3D`

Compute a new 3D Image with slices parallel to a plane of cut, or aligned to the (x,y) plane of a given basis.

The second input is 0 or 1. 1 means "yes" for using a 3D basis. 0 means yesfor using a plane of cut.
Input 3 is the basis: if you fill 0 in the second input, you can let this combo list empty, or with any 3D basis. The plugin will ignore it. If you fill 1 in the second input, you must choose a 3D basis as input 3.
Input 4 is the plane of cut. If you fill 0 in the second input, you must choose a plane as input 4. If you fill 1 in the second input, you can let this combo list

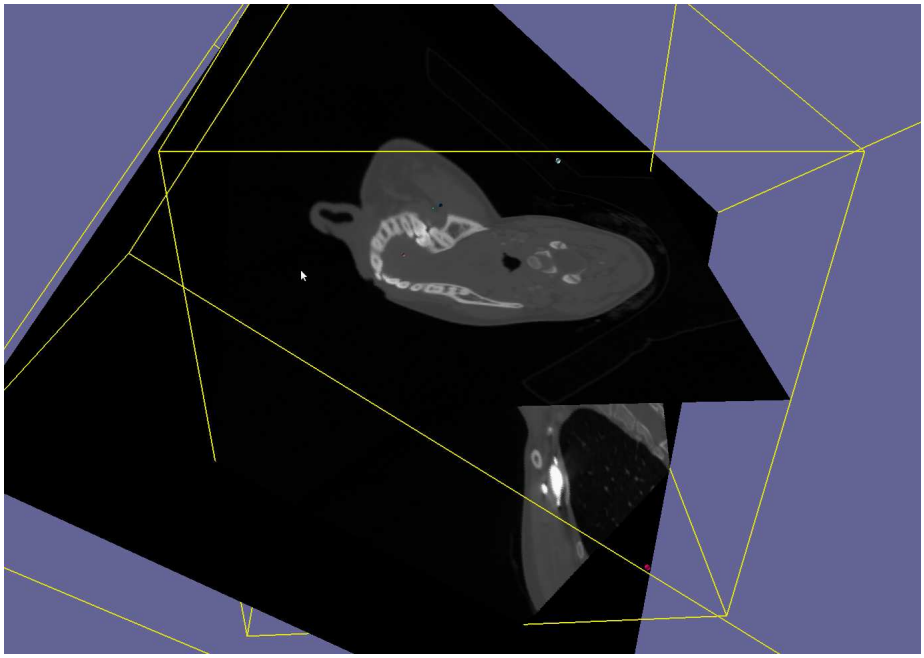

Figure 8: Head resliced

empty, or with any plane. The plugin will ignore it.
Input 5 is the number of cut to process. This will be the slice count of the new 3D image. If you fill with 0, the plugin will compute as many slices as needed for reslicing all the input image.
Input 6 is the space between cuts of the new image. It will be also the new voxel thickness.
Input 7 is a subdivision control. 1 means that only one measurement is done in the original image to compute the grey level of the corresponding voxel in the new image. The position of the measurements are determined randomly for each voxel. A regular sample brings fringes. Random samples will automatically smooth those fringes like a low frequency filter. The computation time is multiplied by this parameter. A good value is between 4 and 8.
Input 8 is the target pixel size. Big values will give less informations, and the image will be less accurate. Small values will give big images and big memory comsuption, and fringes can appear. A value near the voxel size of the original 3D image is a good choice.
If the number of cuts is different from 0, the first slice of the new 3D image will be located exactly mingled to the plane of cuts, or to the (x,y) plane of the 3D basis. All next slices will be computed in the way of the plane normal, or in the way of the Z axis of the 3D basis.
The new 3D image is a resample of original image, so it is recommended to

reslice only original data, and not reslice a reslice. Reslicing an orthoslice is possible, because TIVMI don't resample on orthoslice (see 4.8).
This plugin use multithreads, so comuptation time can be seriously reduced if multithread is available.

4.11 Voxellize a mesh

- Input:
  1. Mesh
  2. Voxel width
  3. Voxel height
  4. Voxel thickness
  5. Bits per pixels (must be 8 or 16)
  6. Margin
- Output:
  1. Name of new 3D image
- Callable name in scripts: `Image3D_Voxellize_Mesh`

Compute a grid of voxels oriented as the world basis. Each slice is parallel to the plane (X,Y). The new image dimension is computed taking into account the bounding box of the mesh and margins. The result is a binary 3D image which depth is specified in input. If the geometric center of a voxel is inside the mesh, the voxel will have a grey level of 255 for 8 bits image, or 65535 for 16 bits image. Else, 0 is stored. If you want to simulate a simple CT Scan acquisition, use the gaussian plug-in to blur the binary image. This plug-in will improved the image since the border of the mesh will be represented as melted grey levels. Be careful, the algorithm is dependant on the mesh's face count. So computation can take a very long time on big meshes.

4.12 Load Analyze Img and Hdr

- Input:
  1. Hdr filename
  2. Big endians
  3. Shift data
- Output:
  1. Name of new 3D image
- Callable name in scripts: `LoadAnalyze`

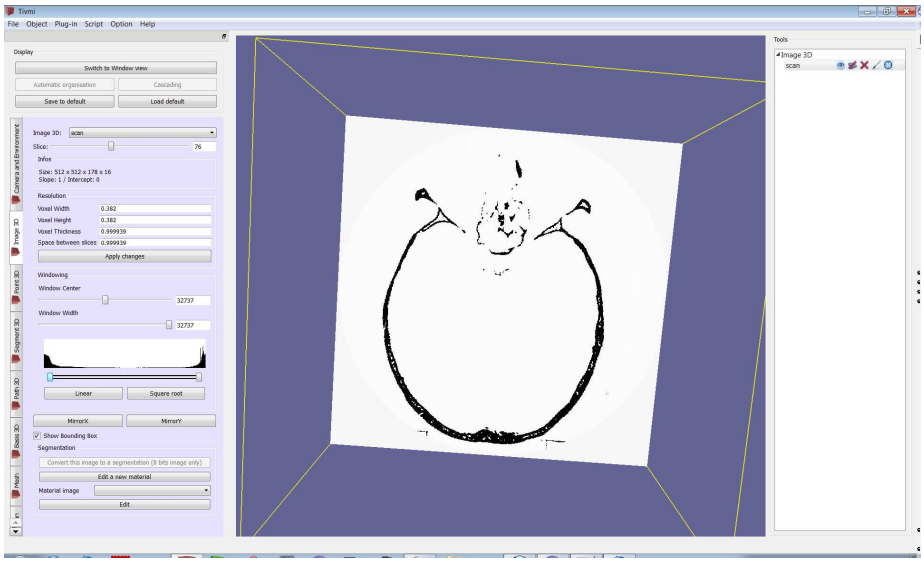

Figure 9: Signed data opened as unsigned data issue

Open the .hdr file given as argument, and automatically deduce the .img filename by replacing .hdr by .img. Setting 0 to the big endians option will suppose that the file has little endian encoding (most of computers). 1 will open file as big endians. Try this option if the opened scan looks like snow.
The shift data option is used to convert signed representation of pixels in the file, to unsigned representation used by TIVMI. In the case of signed data, the black area will look like grey, white will look like black, grey will look like white, and the histogram is splitted (see fig 9). Set 1 to the shift data option to get a correct image.

4.13 Load DICOM

- Input:
    1. Path
- Output:
    1. Name prefix of new 3D image(s)
- Callable name in scripts: `LoadDicom`

Open all files in the given path. To avoid unwanted messages, or troubles, make sure that only DICOM files are present. The plug-in will analyze all DICOM files and try to recognize different studies if available. Thats why the output parameter is a prefix. In fact, if the plug-in detects more than one study,

it will name the studies using the prefix and adding a count. This plugin uses the DCMTK library.

4.14 Load PGM3D

- Input:
  1. Filename
  2. Conform to
  3. Voxel width
  4. Voxel heigh
  5. Voxel thickness
  6. Voxel corner 1 to 4, X, Y and Z
- Output:
  1. Name new 3D image
- Callable name in scripts: `LoadPGM3D`

Open the filename given as first argument. Image dimensions are given in the file. Voxel dimension and 3D positionning are unknown. To setup voxel dimensions and 3D positionning, you can use a 3D image as the second argument, or fill in all following arguments.

4.15 Load PNG serie

- Input:
  1. Path
- Output:
  1. Name new 3D image
- Callable name in scripts: `LoadPNGSerie`

Open all files in the given path. Files are taken in the order given by the system. It is often alphabetically sorted, so be careful with file numbering with no 0 (zero) fillings. e.g. image1.png, image2.png ... image10.png will be opened in this order: 1, 10, 2, 3 etc. Rename to: image001.png ... image010.png. The voxel size is set to 1 by 1 by 1, and is customizable in the TIVMI GUI (see 3.2). Positionning: the first slice is centered on the world origin and parallel to the (X,Y) plane, and next slices are set along the Z axis.

4.16 Load RAW

- Input:
  1. Filename
  2. Size X
  3. Size Y
  4. Size Z
  5. Voxel size X
  6. Voxel size Y
  7. Voxel size Z
  8. Allocated bits
  9. Used bits
  10. Big endians
  11. Shift data
- Output:
  1. Name new 3D image
- Callable name in scripts: `LoadRAW`

Raw file format as no official specification. The simplest way to store a 3D image as raw is to write sequentially all voxels in the file. This plugin opens the file given as first argument. This file is supposed to contains a 3D image in 1 dimension. Voxels are sorted as slices, rows, and finally columns. That mean that the first voxel is located at 1st slice, 1st row, 1st column. The second is located at 1st slice, 1st row, 2nd column, and so on. When a row is complete, the next row begins. When the slice is complete, the next slice begins. Each voxel is stored on "used bits". Typically, it is 8 or 16. Allocated bit is useful for the plugin to allocate a smaller image if necessary: if allocated bits is 16, and used bits is 8, the plugin will generate a 8 bits image, and read the file as 16 bits per voxel. Big endians is a switch to read data as big endian if set to 1 or little endian if set to 0. The shift data option is used to convert signed data to unsigned data. See plugin loadAnalyze (4.12) for more details. Positionning: the first slice is centered on the world origin and parallel to the (X,Y) plane, and next slices are set along the Z axis.

4.17 Load VFF

- Input:
  1. Filename
- Output:

1. Name new 3D image

- Callable name in scripts: `LoadVFF`

Open the VFF file given as first argument. VFF is a file format used by the MicroView software.

5 Creating a Mesh

5.1 Load pts

- Input:

  1. Filename

- Output:

  1. Name of new mesh

- Callable name in scripts: `loadpts`

Open the file given as input. PTS design ASCII file containing a point cloud. This plugin is able to open and detect automatically various ways for describing those points. It will try to read each line as given in the ordered list below. No headings of comments are supported. Each line must end with a CR (carriage return) or LF (line feed) character.

1. X Y Z nX nY nZ r g b
2. X Y Z nX nY nZ c
3. X Y Z r g b
4. X Y Z c
5. X Y Z
6. else: ignore

Where X, Y and Z are floating points numbers designing the coordinates of a vertex in the space.
nX, nY, and nZ are floatting points numbers designing normals. Normals must be normalized.
r, g, and b are integers from 0 to 255 included designing the color of the vertex.
c is an integer from 0 to 255 included designing the grey value. Indexed colors are not supported in TIVMI.

5.2 Load obj

- Input:
    1. Filename
- Output:
    1. Name of new mesh
- Callable name in scripts: `loadobj`

Open the file given as input. TIVMI does not support materials and groups yet.

5.3 Load stl

- Input:
    1. Filename
- Output:
    1. Name of new mesh
- Callable name in scripts: `loadstl`

Open the file given as input. TIVMI supports ASCII and binary formats.

5.4 Load ply

- Input:
    1. Filename
- Output:
    1. Name of new mesh
- Callable name in scripts: `loadply`

Open the file given as input. TIVMI supports only mesh and point cloud formats.

5.5 HMH Surface 2

- Input:
  1. Image 3D
  2. Mode
  3. Gray threshold
  4. Gray min
  5. Gray max
  6. HMH Max length
  7. Binary image
- Output:
  1. Name of new mesh
- Callable name in scripts: `Mesh_HMHSurface2_Image3D`

Creates a mesh from a 3D image using the HMH surface algorithm. The first version of HMH Surface is not supported anymore. Please, use the HMH Surface 2 plugin instead.
Mode defines the way to compute vertices:

- mode=0 : free. Vertices are free to move in spage along the HMH segment. This causes topological troubles on mesh, but HMH computation is respected. Use this setting for simple surfaces.
- mode=1 : limit to voxel. Vertices are free to move inside their own voxel. This corrects topological troubles, but restrict HMH computation. Use this setting for complex surfaces, like trabecular bones.
- mode=2 : no HMH. Vertices are located at the voxel border. There is no HMH computation.

Grey threshold is for automatic threshold on the 3D image. All voxels over this value will be part of the reconstructed mesh. Fill with 0 if you want to use your own binary image (last argument).
Grey min and gray max are thresholds to limit segments for HMH computation. If a segment cross those values, it is stopped. Fill with respectively 0 and $2^{bpp}-1$ where bpp is bit per pixel: e.g. 255 for 8 bits per pixels, or 65535 for 16 bits per pixel.
HMH Max length is a limitation for the HMH segment. Fill with 0 for an automated computation of this value. This will be 10 diagonals of a voxel.
Binary image is taken into account if threshold is 0. All non zero voxels will be used for the surface reconstruction.

5.6 Morph mesh

- Input:
  1. Mesh 1
  2. Mesh 2
  3. Percentage
  4. Tech
- Output:
  1. Name of new mesh
- Callable name in scripts: `Mesh_Morph_Meshs`

This will create a new mesh which vertices position is a ratio from the mesh 1 to mesh 2.
The percentage argument is the ratio to apply: 0 is the mesh 1 position, and 1 is the mesh 2 position.
The tech argument will define how the computation is done:

- 0=index : Mesh 1 and Mesh 2 should have the same number of vertices. The new position of the n^{th} vertex is the ratio applied between the n^{th} vertex of mesh 1 and the n^{th} vertex of mesh 2.
- 1=euclidian : The Mesh 1 is duplicated to create the new mesh, and each vertex is moved to the nearest vertex found on mesh 2. The method can take a while to compute.

6 Creating a 3D Point

6.1 Intersection plane and straight line

- Input:
  1. Plane
  2. Straight line
- Output:
  1. Name of new Point 3D
- Callable name in scripts: `Point3D_Intersection_PlaneStraightLine`

Compute a point at the instersection of a plane with a straight line. If the straight line is parallel to the plane, the plugin will not create the point.

6.2 Orthogonal projection: point on plane

- Input:
  1. Plane
  2. 3D Point
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_Intersection_PlaneStraightLine`

Compute the orthogonal projection of the point to the plane.

6.3 Orthogonal projection: point on straight line

- Input:
  1. 3D Point
  2. Straight line
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_OrthoProjection_PointOnStraightLine`

Compute a point following an orthogonal projection of the point on the straight line.

6.4 Make a 3D point on a 3D segment from a ratio

- Input:
  1. 3D Segment
  2. Ratio
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_Ratio_Segment3D`

Compute the position of a 3D point on the 3D Segment according to a ratio. Ratio is a floatting point number. Zero represents the first point of the segment. 1 represent the second point of the segment. A ratio between 0 and 1 will place the point on the segment proportionnally to the segment length. Less than zero or more than 1 will place the point repectivelly before or after the segment on a virtual straight line including the segment.
e.g.:

- Ratio = 0.5 : the middle of the segment
- Ratio = -1 : the mirror of point 2 compared to point 1
- Ratio = 2 : the mirror of point 1 compared to point 2

6.5 Barycenter path 3D

- Input:
  1. Path 3D
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_Barycenter_Path3D`

Compute the barycenter (i.e. the centroid) of all points defining the given 3D path.

6.6 Shift a point along the normal of a plane

- Input:
  1. 3D point
  2. Plane
  3. length
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_ShiftOnPlaneNormal_Path3D`

Duplicates the point given as first argument and shifts it using the normal of the plane given as second argument. The length between the origin and the new point will be the length given as third argument.

6.7 Skeleton using region growing

- Input:
  1. Image 3D
  2. Start point
- Output:
  1. Name of new 3D Point
- Callable name in scripts: `Point3D_Skeleton_Image3D`

The plugin records the grey value of the pixel containing the start point. This is the grey reference. Then it will look around this pixel on the same slice using a region growing algorithm based on 4 connectivities and will stop when a grey value is different from the grey reference. Then, the mean of all pixels visited is computed, and the point is created.
This plug-in computes a kind of centroid for a given 2D shape. It should be used on a binary image.
For processing multiple slices, see 6.8.

6.8 Skeleton using 2D region growing on all slices

- Input:
  1. Image 3D
  2. Start point
- Output:
  1. Name prefix of new 3D Points
- Callable name in scripts: `Point3Ds_Skeleton2D5_Image3D`

The plugin computes a first skeleton point on the slice containing the given point (see 6.7 for more details).
The skeleton point is shifted to the next slice and to the previous slice, and a new skeleton point is computed. And so on until there is no slice to proceed, or the grey level of the pixel is different from the reference.
Be careful: this plugin does not compute a real skeleton on 3 dimensions. It computes a list of 2D skeletons.

6.9 Convert path3D to points

- Input:
  1. Path 3D
- Output:
  1. Prefix
- Callable name in scripts: `Point3Ds_Explode_Path3D`

Each point of the 3D path is created as an independant 3D point.

6.10 Convert segment to 2 points

- Input:
  1. Segment3D
- Output:
  1. name of first 3D point
  2. name of second 3D point
- Callable name in scripts: `Point3Ds_Explode_Segment`

Create both points of a 3D segment as new 3D points.

6.11 Point3Ds HMH Segment3D HMH value

- Input:
  1. Image 3D
  2. Segment 3D
  3. HMH value
- Output:
  1. name prefix of new points
- Callable name in scripts: `Point3Ds_HMH_Segments3DValue`

Computes a HMH on the 3D segment using the 3D image as reference. If not equal to zero, the HMH value is used. Else, it is computed automatically. The output can be one or more 3D points.

6.12 Intersection between straight line and mesh

- Input:
  1. Straight line
  2. Mesh
- Output:
  1. name prefix of new points
- Callable name in scripts: `Point3Ds_Intersection_StraightLineMesh`

Computes the intersection of a straight line with a mesh as new 3D points. This plugin could be very slow on big meshes.
It is still in developpement: it craches on complex meshes.

7 Creating a 3D path

7.1 Load obj spline

- Input:
  1. Filename
- Output:
  1. Prefix of new 3D path
- Callable name in scripts: `LoadObjSpline`

Opens an obj file containing lines.
The obj file should contain a list of vertices (`v <x> <y> <z>` and a list of lines (`l <vertex_index 1> <vertex_index 2> ...`).

7.2 Orthoprojection Path3D on plane

- Input:
  1. Path3D
  2. Plane
- Output:
  1. Name of 3D path
- Callable name in scripts: `Path3D_OrthoProjection_Path3DOnPlane`

Creates a new 3D path where each point is the corrsponding point in the original 3D path orthogonaly projected on the plane (note that the equidistance, if the 3D path was resampled, will not be respected on the 2D projection). See fig 10.

7.3 Resample Path3D

- Input:
  1. Path3D
  2. Point count
  3. Precision
- Output:
  1. Name of 3D path
- Callable name in scripts: `Path3D_Resample_Path3D`

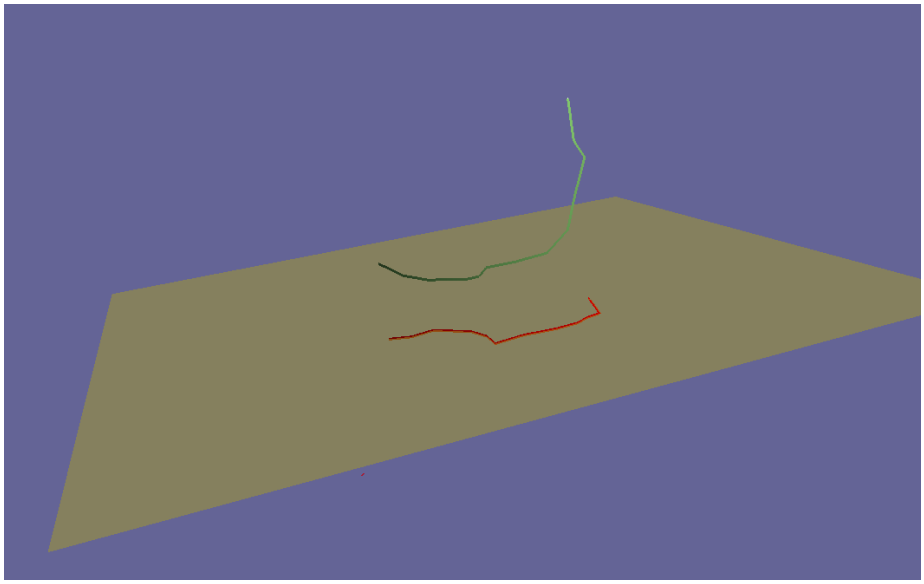

Figure 10: Orthoprojection (red) of a 3D path (green) on a plane.

Creates a new 3D path where distances between two neighboor points is always the same. Point position is interpolated.
The process is iterative until the last point matches the original last point regarding the precision.
It results in a kind of smoothing, and produces equidistant landmarks usable in geometric morphometric analyses. See fig 11.

7.4 Intersection mesh plane

- Input:
  1. Mesh
  2. Plane
- Output:
  1. Prefix name of 3D path
- Callable name in scripts: `Path3Ds_Intersection_MeshPlane`

Creates new 3D paths on the intersection between the plane and the mesh.

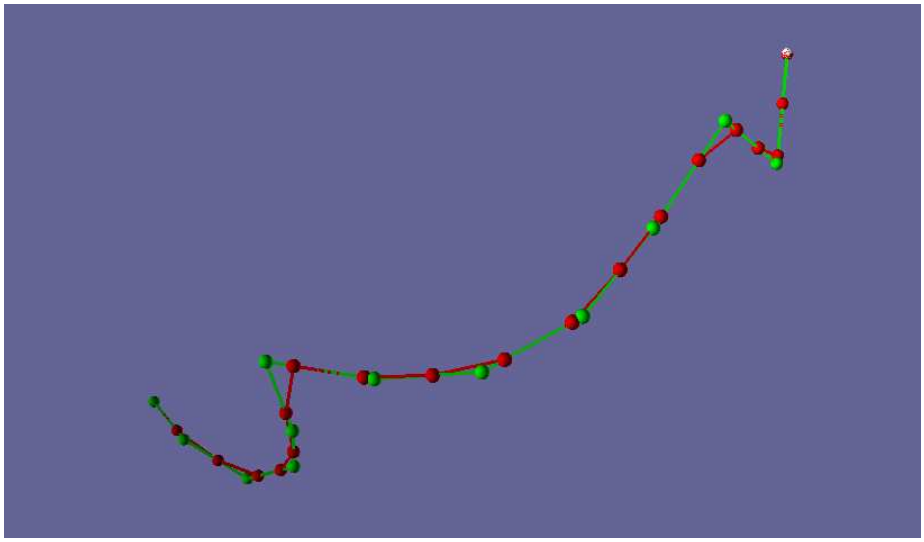

Figure 11: Original path (green) containig 13 points resampled to a path (red) with 20 points.

8 Creating a Plane

8.1 Align on mesh

- Input:
  1. Point 1
  2. Point 2
  3. Point 3
  4. Mesh
  5. Use quadrants
  6. Quadrant plane 1
  7. Quadrant plane 2
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_AlignOnMesh_3Points3D`

This plugin will look for the plane laying on the mesh regarding specific constraints.
The 3 points given as input (named align points) will give to the plugin an idea of the plane to compute (see fig 12). The plugin will start to compute the plane including the 3 align points. Then, it will look for the farthest vertex of the

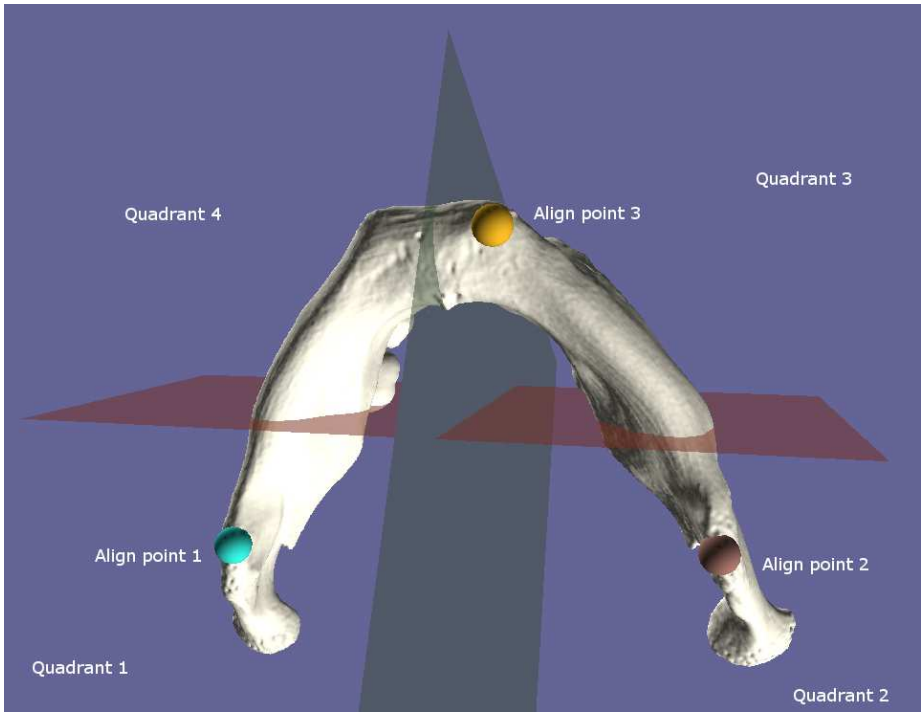

Figure 12: 3 align points defined on a mandibule splitted into 4 quadrants.

mesh on a side of this plane, and the farthest vertex of the mesh on the other side. It will also compute the vertex count on both sides of the plane. Then it will keep the vertex on the side containing the smallest vertex count as a new align point. This new align point will replace one of the 3 align points and for that, it will choose the nearest align point from the new align point.

This is an iterative process, so it is computed until there is no mesh vertex on one side of the plane.
It means that the plane will move in the direction of the smallest part of the mesh.
Quadrants allow to limit movements of align points. It consists in 2 planes that split the mesh into 4 parts. Each point must be in a dinstinct quadrant: the quadrant 1 is defined by the align point 1, quadrant 2 by the align point 2, and quadrant 3 by the align point 3. The last quadrant is quadrant 4 (see fig 12). Align point 1 and align point 2 will move only in its quadrant while the align point 3 will be allowed to switch between quadrant 3 and quadrant 4 (see fig 13).

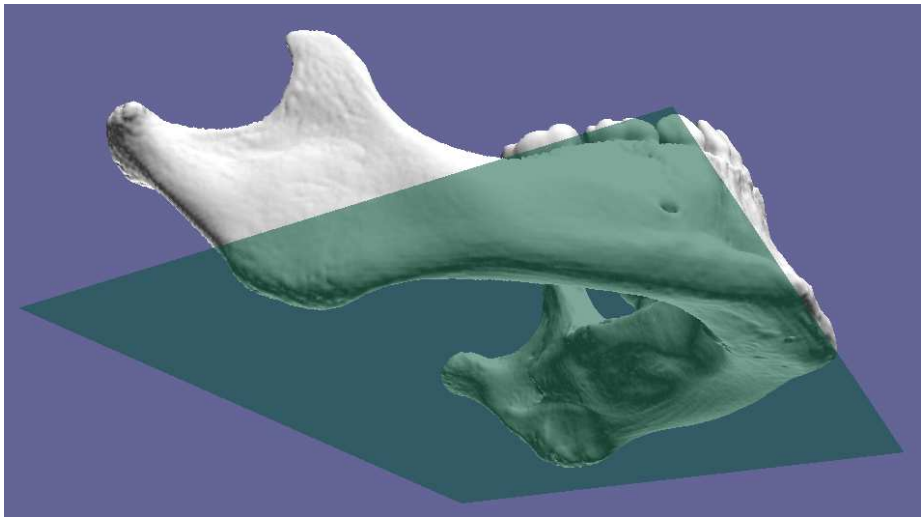

Figure 13: Plane aligned on mandibule.

8.2 Make plane

- Input:
  1. Point 1
  2. Point 2
  3. Point 3
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Construct_3Points`

Creates the plane passing through the 3 points given as input.

8.3 Medium plane

- Input:
  1. Plane 1
  2. Plane 2
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Medium_2Planes`

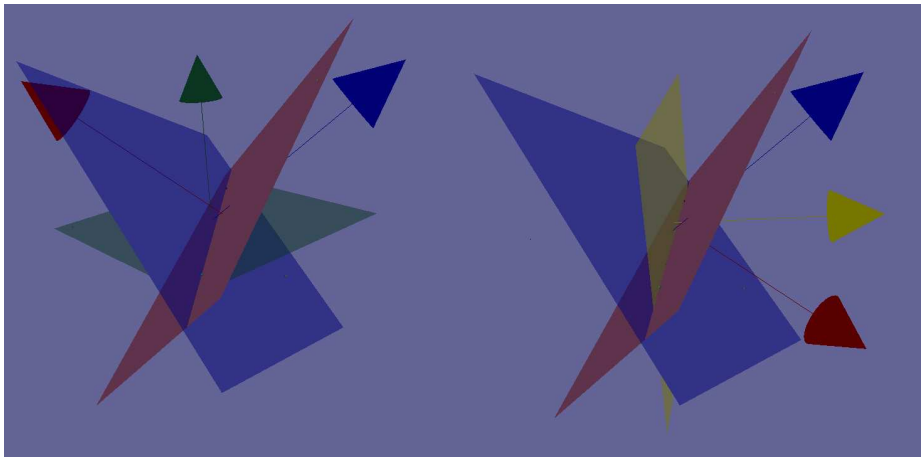

Figure 14: Medium plane computation: the medium plane (green on the left and yellow on the right) is obtained from the blue and red planes. On the right, the red plane normal has been inverted.

Creates the medium (i.e. mean or average) plane between the 2 given planes. The plugin starts to compute the medium normal, and then computes the position of the intersection between the two planes.
So, the normal orientation of input planes is important. If you get an orthogonal plane to the expected plane, try to invert one plane normal and run again the plugin (see fig 14).

8.4 Orthogonal plane from 2 planes through 1 point

- Input:
  1. Plane 1
  2. Plane 2
  3. Point
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Orthogonal_2PlanesThrough1Point`

Creates a plane that is orthogonal to the two planes given as input. Both planes must not be parallel. At this step, there is an infinite number of solutions. The point given as third input will leave only one solution.

8.5 Orthogonal to a plane including a straight line

- Input:
  1. Plane
  2. Straight line
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Orthogonal_PlaneStraightline`

Creates a plane that is orthogonal to the plane given as input and that includes the straight line. That means that all points on the straight line are on the computed plane.
The straight line must not be orthogonal to the plane.

8.6 Orthogonal segment 3D

- Input:
  1. Segment
  2. Point
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Orthogonal_Segment3D`

Creates a plane that is orthogonal to the segment 3D given as input and that includes the point.

8.7 Make an orthogonal point through 2 points

- Input:
  1. Plane
  2. Point 1
  3. Point 2
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Orthogonal_Through2Points`

Creates a plane that is orthogonal to the plane given as input and that includes both points.
Both points must not describe an orthogonal segment to the input plane.

8.8 Parallel to straight line through 2 points

- Input:
  1. Straight line
  2. Point 1
  3. Point 2
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Parallel_StraightLine2Points`

Creates a plane that is parallel to the straight line given as input and that includes both points.
Both points must not describe a parallel segment to the straight line.

8.9 Shift plane to a 3D point

- Input:
  1. Plane
  2. 3D Point
- Output:
  1. Name of plane
- Callable name in scripts: `Plane_Shift_Point`

Creates a plane that is parallel to the given plane and that includes the point.

8.10 Bench vice a mesh

- Input:
  1. Mesh
  2. Straight line
- Output:
  1. Prefix for 3D planes
- Callable name in scripts: `Planes_BenchVice_StraightlineMesh`

Creates 2 planes like a bench vice where the screw is the straight line (see fig 15). This plugin also creates the 2 contact points. This is usefull to compute the distance between the two planes.

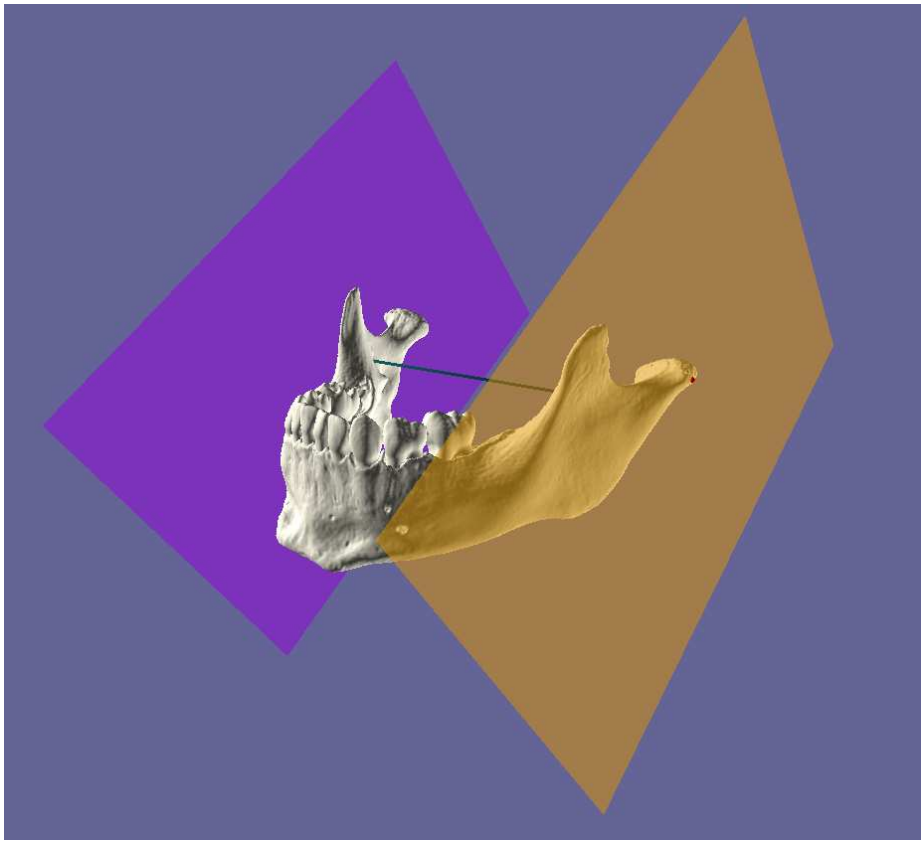

Figure 15: Making a bench vice around a mesh.

8.11 Constrained Bench vice a mesh

- Input:
  1. Mesh
  2. Ground plane
  3. Parallel straight line
  4. top limit
  5. bottom limit
- Output:
  1. Prefix for 3D planes
- Callable name in scripts: `Planes_ConstrainedBenchVice_PlaneStraightLineMesh`

Creates 2 planes like a bench vice. Both planes are orthogonal to the "ground plane" and parallel to the straight line (see fig 16). The contact point is taken only if its distance to the ground plane is less or equal to the top limit on the same side as the ground plane normal, and less or equal to the bottom limit on the ground plane on the other side.
For limit inputs, you can set 0 if you want to take into account only one side of the ground plane, or -1 if you want to ignore limitation.
Both contact 3D points are also created.
Be careful: created planes are infinite planes as all planes in TIVMI. If you use the limit inputs, you mean that you want to create half planes or a kind of stripe plane, but TIVMI will display them as full infinite planes.

8.12 Bench vice a limited mesh

- Input:
  1. Mesh
  2. Straight line
  3. Plane 1
  4. Plane 2
- Output:
  1. Prefix for 3D planes
- Callable name in scripts: `Planes_LimitedBenchVice_StraightlineMeshPlanes`

Creates 2 planes like a bench vice as the plugin bench vice (see 8.10), but only vertices on the same side of the plane1 normal and on the same side as plane 2 normal are tested.
This is the same as the bench vice plugin on a sliced mesh.

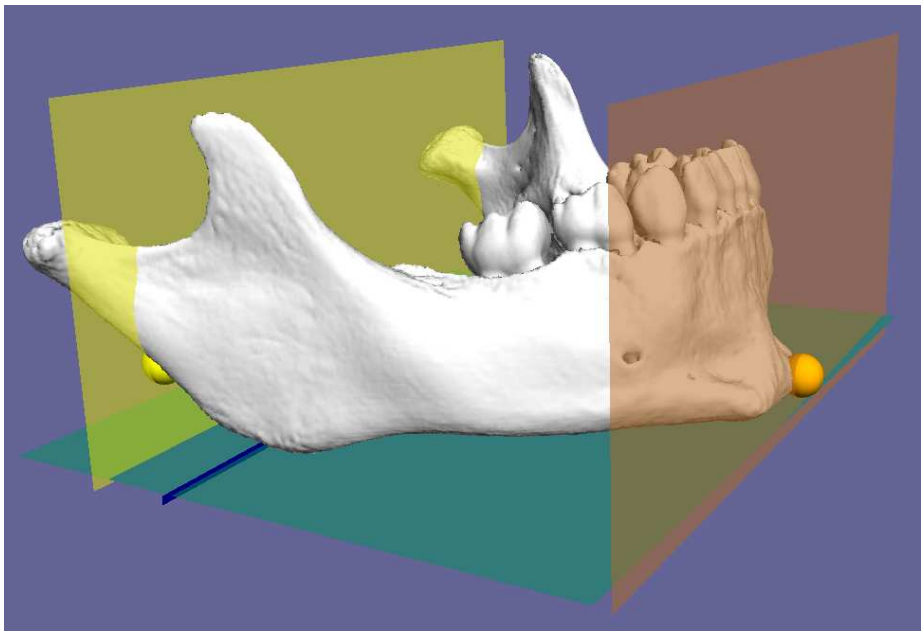

Figure 16: Constrained bench vice on a mandibule: top limit to 1 cm, and bottom limit to 0 (ignore mesh under the ground plane). Impact points are highlighted.

9 Creating a Segment 3D

9.1 Make 3D segment from 2 points

- Input:
  1. Point 1
  2. Point 2
- Output:
  1. Name of 3D segment
- Callable name in scripts: `Segment3D_Construct_3Points`

Makes a new 3D segment where the starting point is point 1 and the ending point is point 2.

9.2 Maximum distance from a 3D path to a plane

- Input:
  1. Path 3D
  2. Plane
- Output:
  1. Name of 3D segment
- Callable name in scripts: `Segment3D_MaxDistance_Path3DPlane`

Makes a new 3D segment where the starting point is a point from the 3D path, and the ending point is the orthogonal projection of the starting point on the plane. The chosen point on the 3D path is the farthest one from the plane.

9.3 Minimum distance between 2 path3Ds

- Input:
  1. Path 3D 1
  2. Path 3D 2
- Output:
  1. Name of 3D segment
- Callable name in scripts: `Segment3D_MinDistance_2Path3Ds`

Makes a new 3D segment where the starting point is a point from the first 3D path, and the ending point is a point from the second 3D path, and where the segment length is the smallest of all possible combinations.

9.4 Shift along plane normal with ratio

- Input:
  1. 3D Segment
  2. Plane
  3. Ratio
- Output:
  1. Name of 3D segment
- Callable name in scripts: `Segment3D_Shift_Segment3DPlaneNormalRatio`

Makes a new 3D segment parallel to the given one, moved in the same direction as the plane normal, at the distance of the specified input ratio (the ratio is applied to the normalized normal).

9.5 Segment angle spread

- Input:
  1. Point 1 : axis
  2. Point 2: from
  3. Point 3: to
  4. Angle step in degrees
  5. Number of segments (from and to included)
- Output:
  1. 3D segment prefix
- Callable name in scripts: `Segment3Ds_AngleSpread_3Points`

Makes a serie of segments where the starting point is always the point 1. Ending points are spread between point 2 and point 3.
In the case of an angle step, un unknown number of 3D segments are created and the angle with the previous segment is always the same. The one but last 3D segment may touch the point 3 and the last segment will, but the angle will not be respected (see fig 17).
In the case of a number of segments, the angle with the previous segment is unknown but always the same. The last segment will touch the point 3. This number should be greater than 2.
In all cases, all segments are included into a plane and have the same length computed as the biggest length between the point 1 to point 2 segment and the point 1 to point 3 segment.
This plugin can be used for the computation of a bisector of an angle.

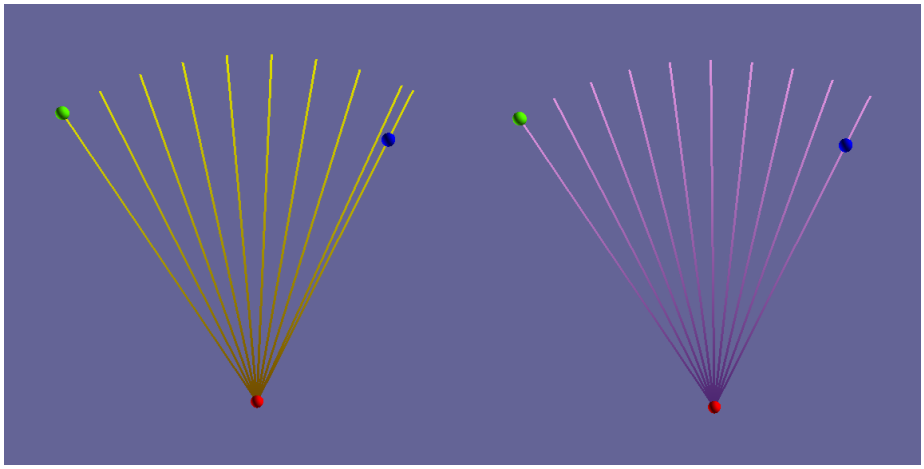

Figure 17: Segment 3D angle spread plugin: difference between setting up angle (left: 7 degrees) and segment count (right: 10).

9.6 Segment ladder

- Input:
  1. Point 1 : summit
  2. Point 2: base
  3. Point 3: base
  4. Number of segments (base and summit included)
- Output:
  1. 3D segment prefix
- Callable name in scripts: `Segment3Ds_Ladder_3Points`

Makes a serie of segments parallel to the base segment (point 2 to point 3 segment) contained in the plane including the 3 points.
The distance with the previous segment is unknown but always the same. The first segment is the base, and the last includes the summit (see fig 18).

10 Creating Data

10.1 Angle 2 planes

- Input:
  1. Plane 1
  2. Plane 2

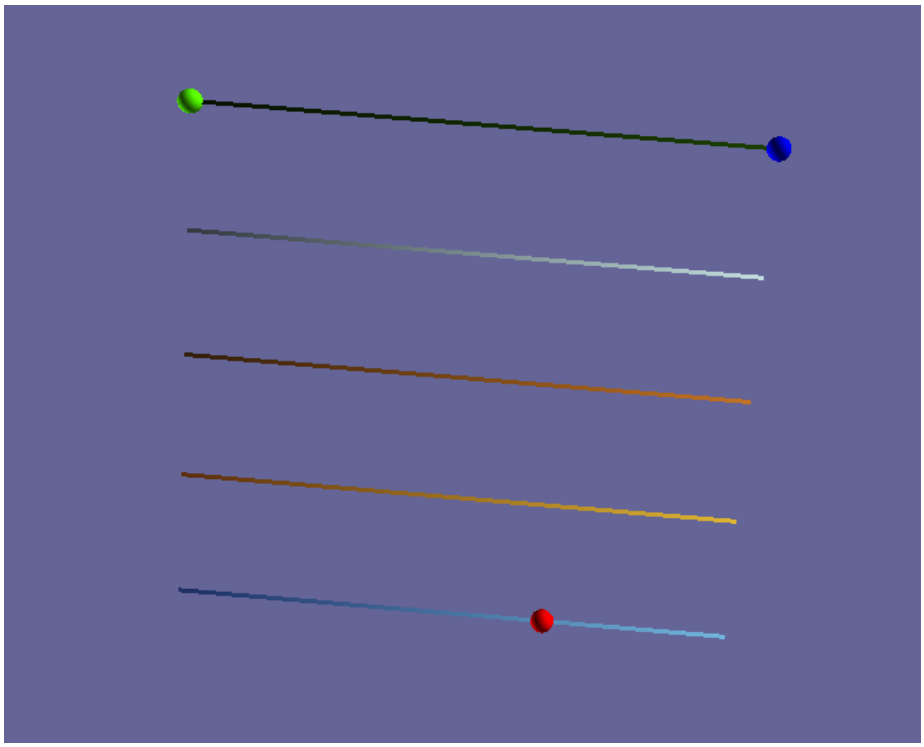

Figure 18: Segment 3D ladder example.

- Output:
    1. Name of angle
- Callable name in scripts: `Data_Angle_2Planes`

Computes the angle between 2 planes in degrees. There are 2 solutions. The plugin computes the angle between the two normals, so only one solution is found. To find the other solution, you can invert the normal of a plane, or compute manually the complementary angle to 180 degrees.

10.2 Angle 3 points

- Input:
    1. Point 1 (angle)
    2. Point 2
    3. Point 3
- Output:
    1. Name of angle
- Callable name in scripts: `Data_Angle_3Points`

Computes the angle between 3 points where the point given as first input is the summit of the angle.

10.3 Height of triangle

- Input:
    1. Point 1 (summit)
    2. Point 2 (basis)
    3. Point 3 (basis)
- Output:
    1. Name of height
- Callable name in scripts: `Data_HeightOfTriangle_3Points`

Computes the height of a triangle where the point given as first input is the summit of the triangle.

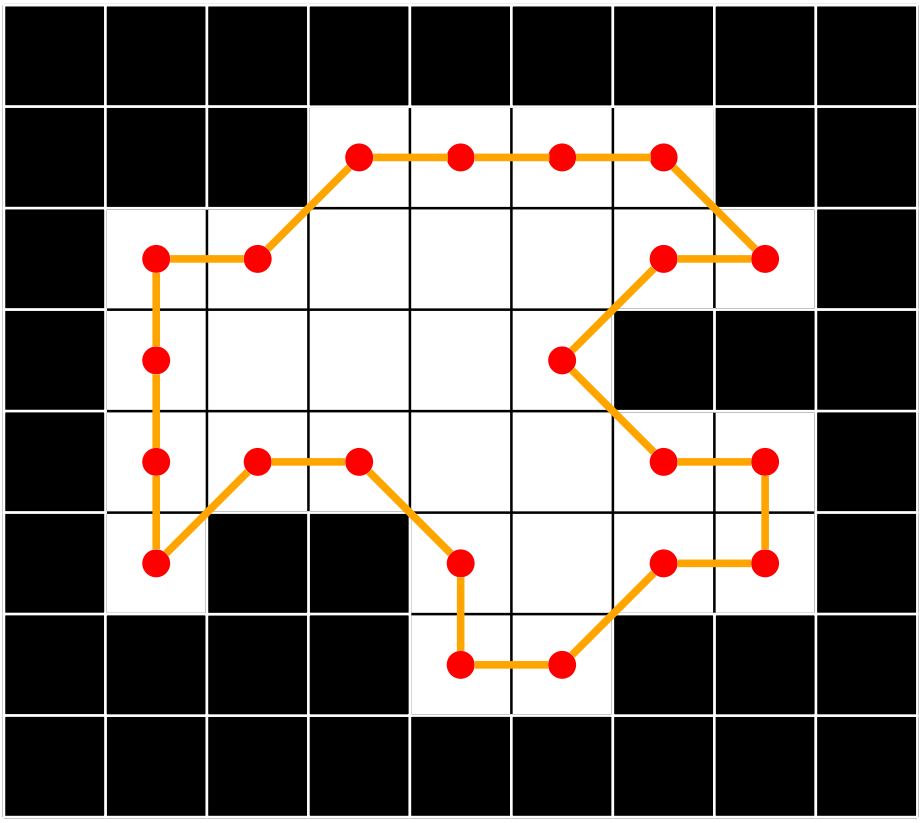

Figure 19: Perimeter computation of a 2D shape.

10.4 Perimeter of a shape in a binary slice

- Input:
  1. Binary Image3D
  2. Point3D
- Output:
  1. Name of perimeter
- Callable name in scripts: `Data_Perimeter_Image3DPoint3D`

Computes the perimeter of a shape. The 3D point gives the slice to proceed and must target the shape. The plugin will start from this point, and look for the shape border. When found, it will follow the 8-connectivity border of the shape. The final length is the sum of segments linking the center of pixels inside the shape (see fig 19).

10.5 Volume from voxels

- Input:
  1. Image 3D
  2. Grey value from
  3. Grey value to
- Output:
  1. Name of volume
- Callable name in scripts: `Data_VolumeVoxels_Image3D`

Computes the volume as the sum of all voxel volumes which grey value are between the specified grey value from (included) and the grey value to (included).

10.6 Weight of all materials in a 3D image

- Input:
  1. Original 3D image
  2. Segmented 3D image
  3. Slope (from dicom tags)
  4. Intercept (from dicom tags)
  5. factor
- Output:
  1. Prefix name of data to output
- Callable name in scripts: `Data_Weight_MaterialImage3D`

This plugin is avaliable only on segmented images. You can convert a 8 bits image to a segmented image by clicking the "convert to segmentation" button in the Image3D object control window (group box segmentation) (see 3.2). Note that your image must be shown and must be a 8 bits depth image.
A material is a set of voxels with the same grey level in the segmented image. For example, 0 (black) is the air, 1 is the soft tissues, and 2 is the bone.
The plugin will look at the grey value of each voxel in the original 3D image and use the Slope and Intercept inputs to compute the Hounsfield value of each voxel:
$Voxel_hounsfield = \frac{grey_value * Slop}{Intercept}$
You can find the slope and intercept values in TIVMI in the user interface of 3D image (see 3.2). This information are provided only if the 3D image has been loaded by the load dicom plugin.
Then the weight of the voxel is computed:

$Voxel_weight_cm3 = \frac{Voxel_hounsfield}{1000.0} + 1.0$
This weight is given for a 1 cm^3 voxel. It is scaled to the real volume of a voxel:
$Voxel_weight = Voxel_weight_cm3 * \frac{Voxel_Width * Voxel_Height * Distance_between_slices}{1000.0}$
Then all voxel weights are summed to get the material weight.
The factor is used in the case of scaled grey levels on the original 3D image: if the original data was recorded on 12 bits, and the 3D image was scaled to 16 bits, use 16 as factor to divide the grey level by 16 before the weight computation.

BE CAREFUL: this plugin works fine only if the slope and intercept are correctly provided by the scanner manufacturer in the Dicom file. Our experimentation shows that it is never the case! In fact, the reconstruction method of images changes the Hounsfield scale and the slope and intercept are not updated.

11 Creating Straight Lines

11.1 Make straight line from 2 planes

- Input:
  1. Plane 1
  2. Plane 2
- Output:
  1. Name of straight line
- Callable name in scripts: `StraightLine_Construct_2Planes`

Creates the straight line at the intersection of both specified planes. Planes must not be parallel.

11.2 Make straight line from 1 segment

- Input:
  1. Segment
- Output:
  1. Name of straight line
- Callable name in scripts: `StraightLine_Convert_Segement`

Creates the straight line containing the segment.

11.3 Shift a straight line to a 3D point

- Input:
  1. Straight line
  2. Point 3D
- Output:
  1. Name of straight line
- Callable name in scripts: `StraightLine_Shift_Point`

Creates the straight line parallel to the given one, and containing the 3D point.

12 Output to file

12.1 Reslice and save to radio tiff a 3D image

- Input:
  1. Image 3D
  2. Plane
  3. Pixel dimension
- Output:
  1. filename
- Callable name in scripts: `ResliceAndSaveToRadioTiff`

This plugin was provided to save memory when computing a reslice to be saved as radio tiff.
Now we suggest you to reslice your 3D image using the reslice plugin (see 4.10) and use the save to radio tiff plugin (see 12.10).

12.2 Project each point of a mesh into an image

- Input:
  1. Mesh
  2. Plane
  3. Thickness
  4. Pixel size
  5. Point diameter

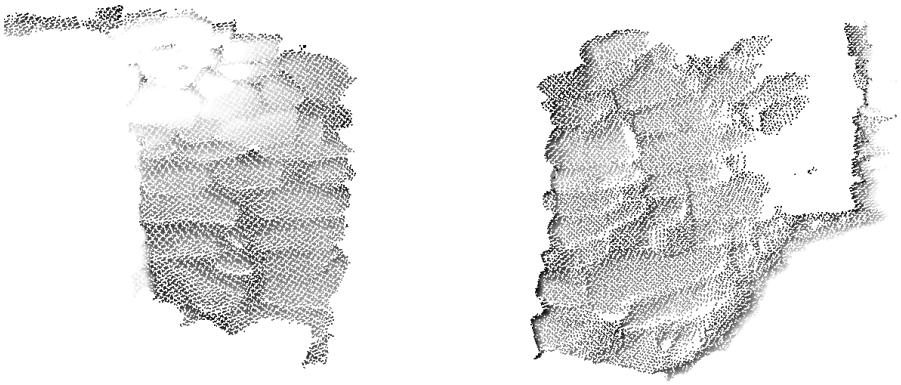

Figure 20: Point cloud projection on 2D image.

- Output:
  1. filename
- Callable name in scripts: `SaveMeshProjectionToPng`

This plugin is used to perform an orthoprojection of a point cloud. The plane represents the 2D image. The thickness the the volume from the plane to a distance of "thickness" where points are taken into account.
The 2D image has white background. The more the point are close to the plane, the more it is black, and the more the point are far, the more it is white.
The pixel size is the real size of the new image pixel in the 3D world. You should choose a value near the distance between 2 neighboor mesh points.
The resulting image looks like embossed (see fig 20). Measurements are possible because you know the pixel size.
The point diameter is ignored.

12.3 Save Path 3D to Svg

- Input:
  1. Path 3D or group

2. Xup:1 Yup:2 Zup:3 auto:0

- Output:
  1. filename
- Callable name in scripts: `SavePath3DToSvg`

Svg is a 2D vectorized file format. This plugin is useful to import TIVMI path 3D into graphic software.
The 3D path or a group of 3D paths should be included in a plane or several parallel planes. The second argument will force the path to be oriented with a chosen up axis. The up axis will be the canvas height in the svg. Auto will let TIVMI determine the best orientation. In all cases, the 3D Path is projected on a plane and exported as a 2D shape.

12.4 Save Image3D to analyze img and hdr files

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToAnalyzeImg`

Save a 3D Image to the Anayze file format. This format will lose the image 3D position in the 3D space, and additional information read in DICOM for example.

12.5 Save Image3D to amira mesh file format

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToImage3DAmiraMesh`

Save a 3D Image to the Amira (= Avizo) mesh file format (.am). This format will lose the image 3D position in the 3D space, and additional information read in DICOM for example.

12.6 Save segmented Image3D to amira mesh file format

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToImage3DLabelsAmiraMesh`

Save a 3D Image to the Amira (= Avizo) mesh file format (.am). The image must have a 8 bits depth. This format will lose the image 3D position in the 3D space.

12.7 Save mesh to obj

- Input:
  1. Mesh
- Output:
  1. filename
- Callable name in scripts: `SaveToObj`

Save a mesh to the .obj file format (wavefront).

12.8 Save path 3d to obj

- Input:
  1. Path 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToObjSpline`

Save a 3D path to the .obj file format (wavefront). It consists in a list of vertices ($v < x >< y >< z >$) followed by a line ($l < vertex_index >< vertex_index > [vertex_index]...$).
This is usefull to load a 3D path in Autodesk 3DSMax for example.

12.9 Save mesh to ply

- Input:
  1. Mesh
  2. Binary
- Output:
  1. filename
- Callable name in scripts: `SaveToPly`

Save a mesh to the .ply file format (stanford). It can output binary (set to 1) or ASCII (set to 0).

12.10 Save Image 3D to one radio tiff file

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToRadfioTiff`

Collapses all slices of the 3D image to one and averages all grey levels. It results in a radiography-like 2D image. The tiff file format allows to keep the 16 bits depth if needed.

12.11 Save Image 3D to tiff

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToTiff`

Save each slice into a tiff file. The tiff file format allows to keep the 16 bits depth if needed. If pixels are not square (see 4.8), the plugin will resample each slice to fit to square pixels. The pixel dimension will be the smallest dimension of the original pixel.

12.12 Save an object to tso

- Input:
  1. Object
- Output:
  1. filename
- Callable name in scripts: `SaveToTso`

The tso file format is for "Tivmi Serialized Object". It consists in a binary version of the object, writable in a file. The object type is stored in the file, so TIVMI is able to read it and create a well typed object (see 13.1).
This plugin is used by the "save state" menu.

12.13 Save Image 3D to VFF

- Input:
  1. Image 3D
- Output:
  1. filename
- Callable name in scripts: `SaveToVFF`

Save a 3D image to the VFF file format. VFF is a file format used by the MicroView software.

13 Creating any object

This section contains plugins that are able to create any kind of object in TIVMI. For the moment, there is only one of those.

13.1 Load tso

- Input:
  1. Filename
- Output:
  1. Name of object to create
- Callable name in scripts: `LoadTso`

The tso file format is for "Tivmi Serialized Object". It consists in a binary version of the object, writable in a file. The object type is stored in the file, so TIVMI is able to read it and create a well typed object.
This plugin is used by the "Load state" menu.

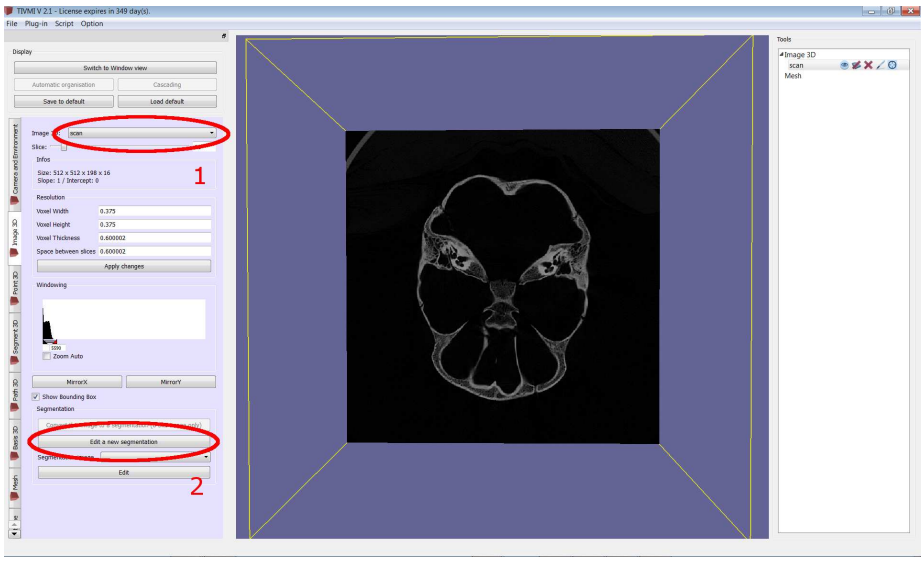

Figure 21: Create a new segmentation

Part V
Segmentation

The segmentation consists in labelling voxels of a 3D image. A label is a number between 1 and 255. All voxels with the same label is a part of the same material (or region). A voxel cannot be tagged by 2 or more labels. The label 0 (zero) is used for non tagged voxels. The segmentation is stored as a 8 bits depth 3D Image. It can be displayed by TIVMI in the 3D view as a grey-level 3D image like other 3D images, but a specific interface is provided to create and edit a segmentation. Each label can be named and has a color.
A segmentation is always associated to a 3D image: the parent image. This parent image is a 8 bits or 16 bits 3D image like a CT-Scan.

14 Creating a segmentation

14.1 From scratch

To create a segmentation, you need the parent image to be displayed in the 3D view. Then, in the Object Control pannel, in the Image 3D tab (see 3.2), select the parent image in the list (fig 21 step 1), and click on the button “Edit a new segmentation” (fig 21 step 2).
A new 8-bits 3D image si created with the same dimensions and position in 3D as the parent image. The name is automatically set to the parent image name

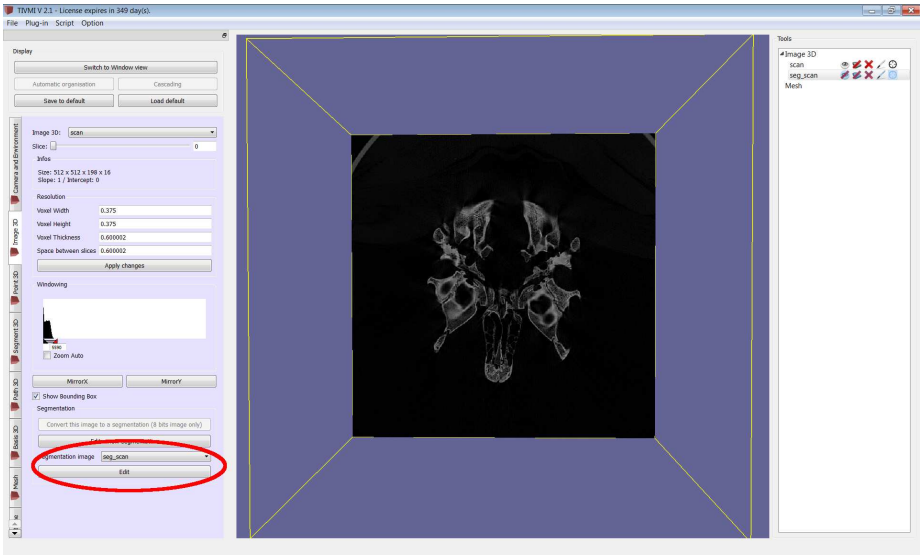

Figure 22: Create a segmentation from a 8 bits 3D image

with the "seg_" prefix.

14.2 From a 8 bits 3D image

If you own a 8-bits 3D image imported from another software, you can convert it to a segmentation. Display the 3D image in the 3D view, then choose it in the image list, and click on "convert this image to a segmentation (8 bits image only)". The 3D image will be tagged as a segmentation image, and will appear in the segmentation list below.

15 Editing a segmentation

To edit a segmentation, the parent image must be displayed in the 3D view. in the Object Control pannel, in the Image 3D tab (see 3.2), select the parent image in the list. In the segmentation box, a list of potential segmentation is available. This list is filled with 8-bits 3D image which have the same dimension as the parent image. Choose a segmentation in this list, then click on "Edit" (fig 22).

16 The segmentation GUI

The GUI starts with a default placement: the material list is on the left (1 on fig 23). The slice view associated with the image histogram is centered (2 on fig 23), and tools are on the right (3 on fig 23).

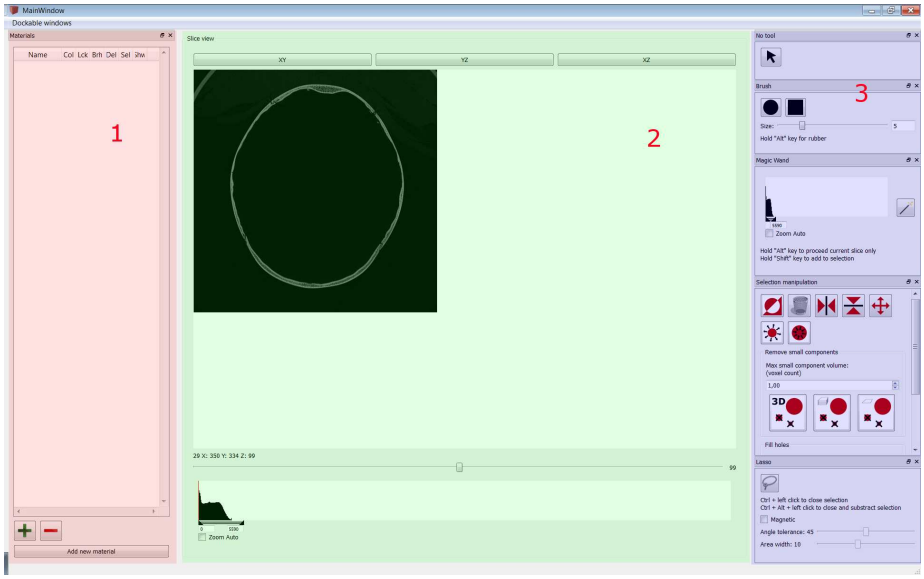

Figure 23: Default configuration of segmentation GUI

The Material list and tools are dockable windows, and can be moved out of the segmentation window. This is useful to maximize the slice view surface on dual screens (see fig 24).

The main goal of this GUI is to help you to set a label on voxel. This means set a voxel as a material. It consists in selecting voxels with tools, and add the selection to a material. You can select any voxels even if they are labelled or not.

16.1 Slice view

This part of the GUI displays the parent image with the segmentation as an overlay. 3 buttons on the top change the slice orientation. XY is the default 3D Image slice orientation. YZ and XZ are orthoslices. As in the 3D view of TIVMI, non isometric voxels are displayed as rectangular pixels if needed (see fig 25).

The slider below let you navigate in slices.

The histogram modulates the windowing of the parent image, and works as the histogram in the Object Control pannel / Image 3D tab.

Mouse controls:

- pointing: move the mouse on the slice (1 in fig 26) and see below the slice the grey level and 3D coordinates of the voxel targeted (2 in fig 26). A vertical red line shows the grey level in the histogram in real time (3 in fig 26).

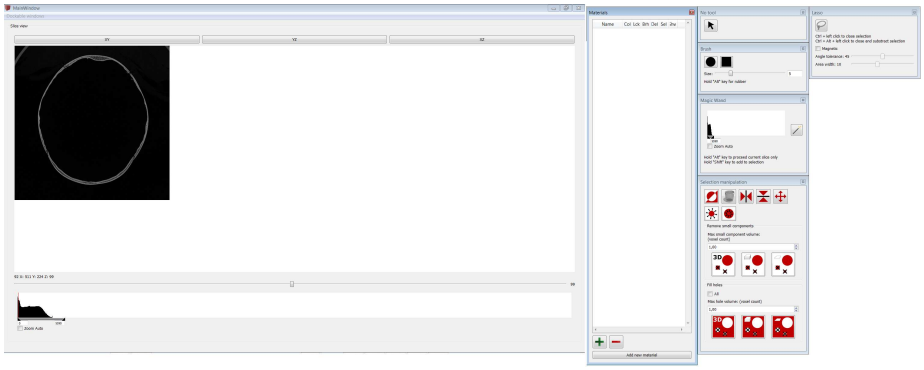

Figure 24: Configuration of segmentation GUI customized for dual screens

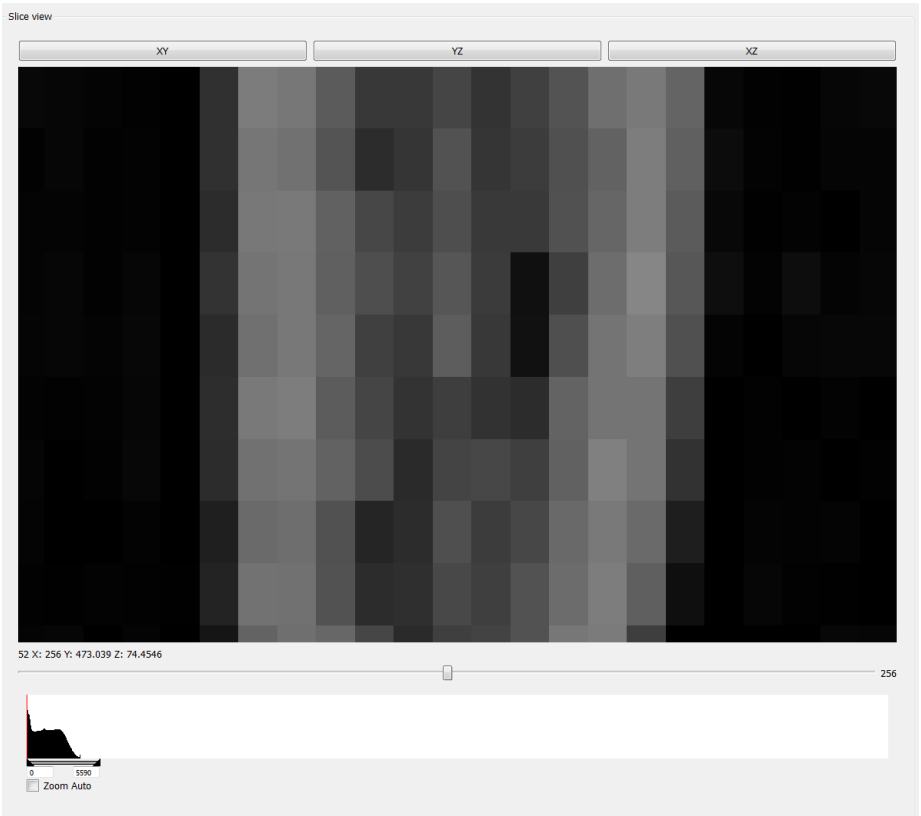

Figure 25: Rectangular pixels on the segmentation slice view.

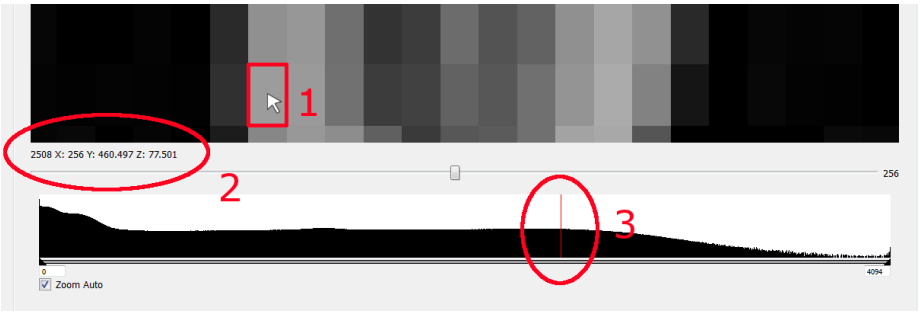

Figure 26: Voxel informations in the segmentation slice view.

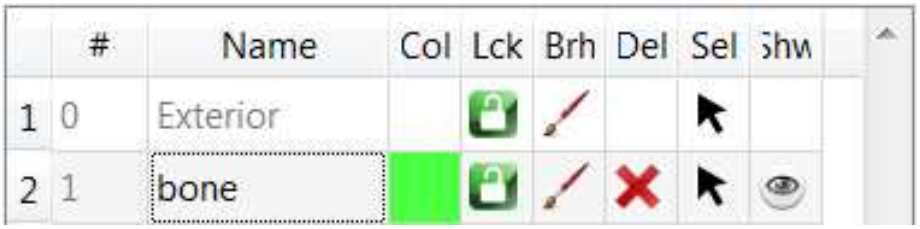

Figure 27: Material list for segmentation.

- Hold right button and move up or down to zoom or unzoom. TIVMI will center the zoom on the mouse pointer.
- Hold middle button and move up, down, left or right to move the slice.
- The left button is used by tools

16.2 Materials list

This list contains materials. Each material is a label from 0 to 255 and this value is shown in the first column.
See fig 27 and 28 for columns details.

- The first column is the label number of the material. This is not editable.
- The second column is the name of the material. Double click on it to edit the name.
- The third column is the color of the material displayed in the slice view. Simple click on it to change the color.
- The fourth column is used to lock the material. More details about this feature is detailed below.

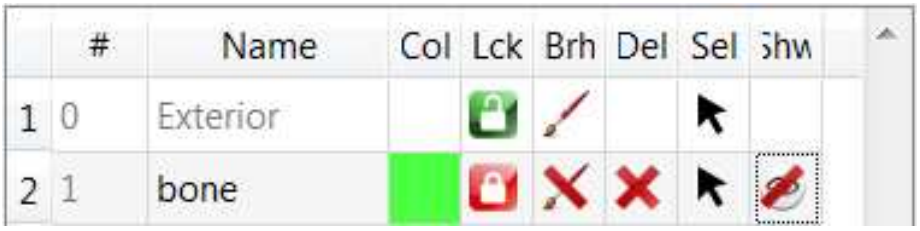

Figure 28: Material list for segmentation with locks.

- The fifth column is a brush lock. If set (pencil crossed), tools will not have effect on this material.
- The sixth column is used to delete the material. All voxel with this material label will be set to label 0 (zero).
- The seventh column will select all voxels of the material.
- The eighth column will hide or show the material in the slice view.

The material with the label 0 is a special material. Its name is always "exterior". It can't be renamed, deleted. It has no color and can't be shown in the slice view. But it can be locked, brush locked, and selected.
The green "+" button will transfer the selection to the selected material. You must first select a material in the material list. Each selected voxel will be labelled with the chosen material. If a selected voxel is already labelled by another material, and if this material is locked, then the voxel will keep its label and will not be transferred to the chosen material.

Fig 29 and 30 show how a selection (in red) crossing the skull material (in green) and the endocranium material (in blue) is added to a 3rd material named "test" (in light purple) while skull is locked and endocranium is unlocked. The skull in green is not modified.

Fig 31 and 32 show how a selection (in red) crossing the skull material (in green) and the endocranium material (in blue) is added to a 3rd material named "test" (in light purple) while skull is unlocked and endocranium is locked. The endocranium in blue is not modified.

Fig 33 and 34 show how a selection (in red) crossing the skull material (in green) and the endocranium material (in blue) is added to a 3rd material named "test" (in light purple) while skull and endocranium are locked. Neither the skull nor the endocranium are modified.

The red "-" button will set all selected voxels to the label 0 (zero). As the "+" button, locked materials are not affected. No material needs to be selected before.
The "Add new material" button will create a material with a generic name and a random color.

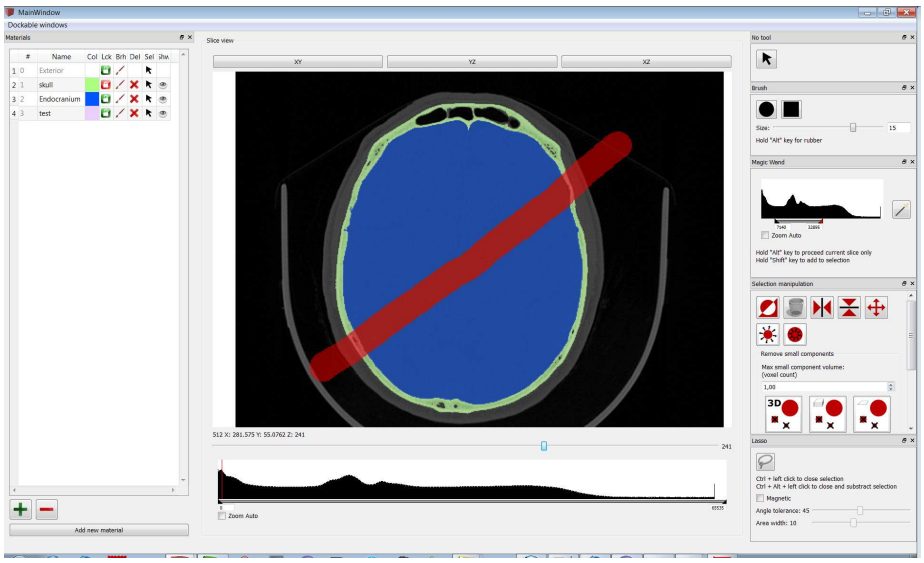

Figure 29: Material lock scenario 1: selection crossing skull and endocranium

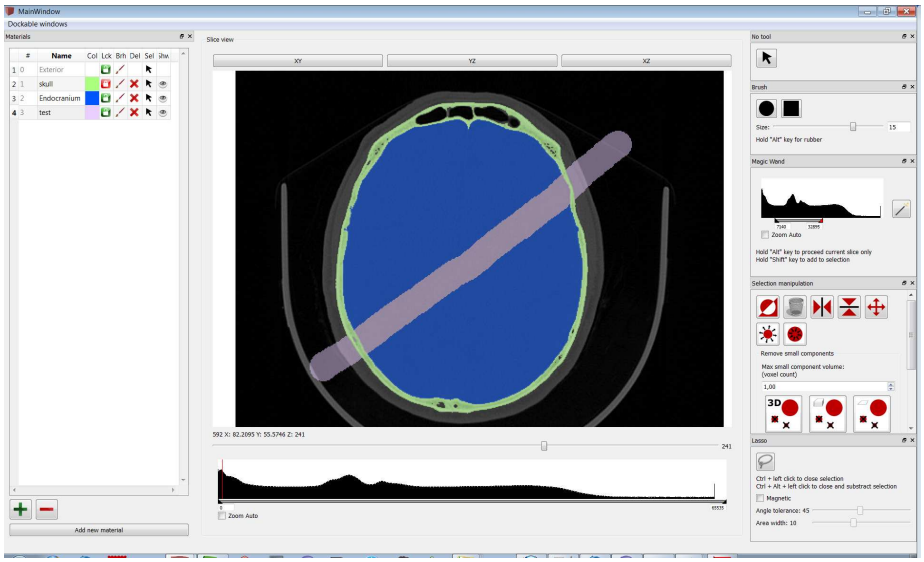

Figure 30: Material lock scenario 1: selection crossing skull (locked) and endocranium (unlocked) added to the "test" material

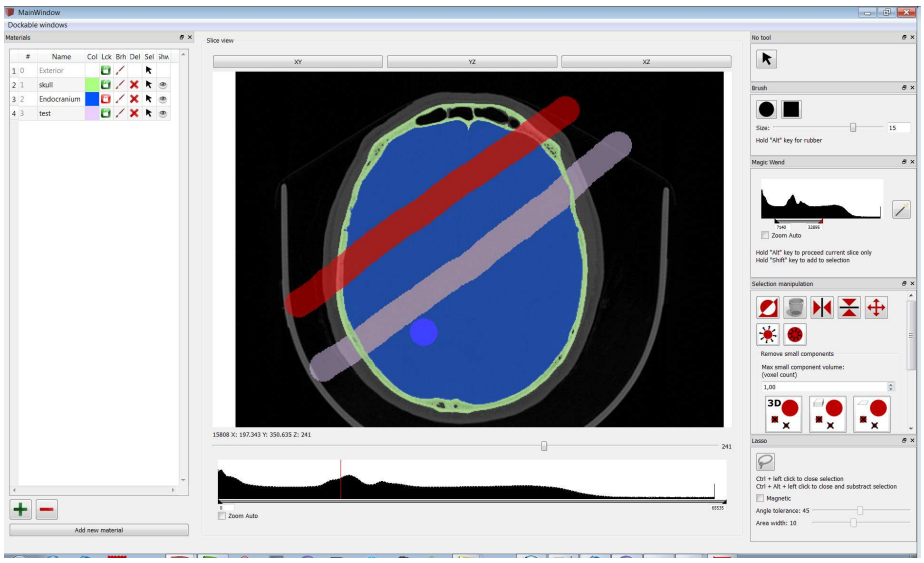

Figure 31: Material lock scenario 2: selection crossing skull and endocranium

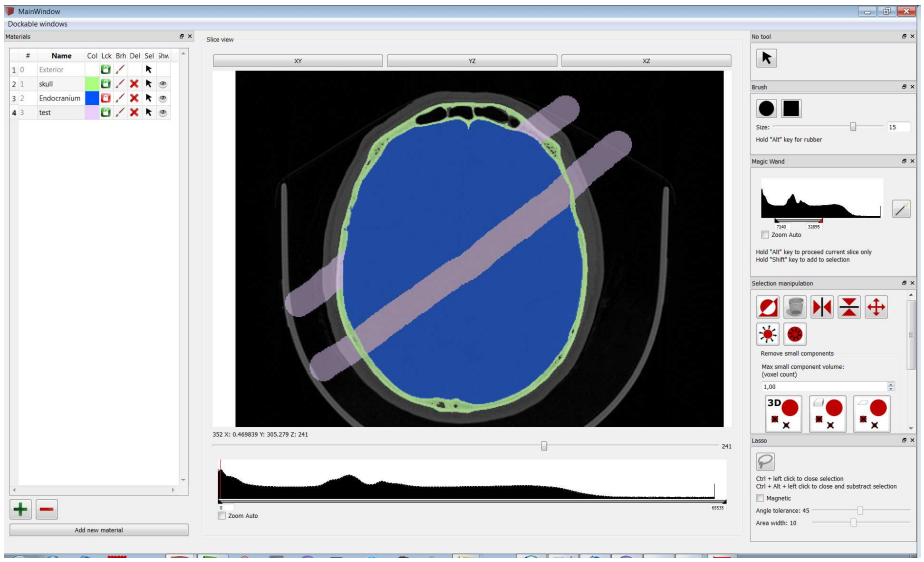

Figure 32: Material lock scenario 2: selection crossing skull (unlocked) and endocranium (locked) added to the "test" material

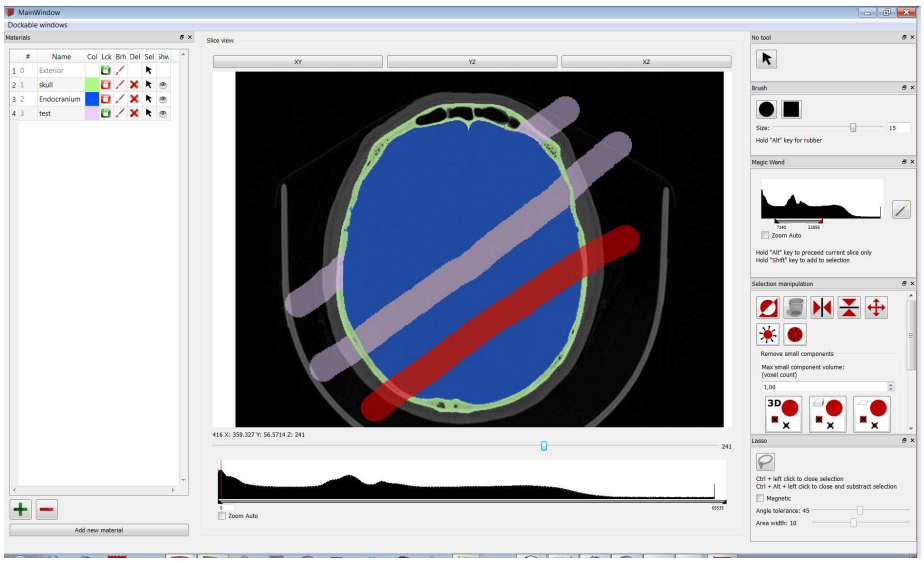

Figure 33: Material lock scenario 3: selection crossing skull and endocranium

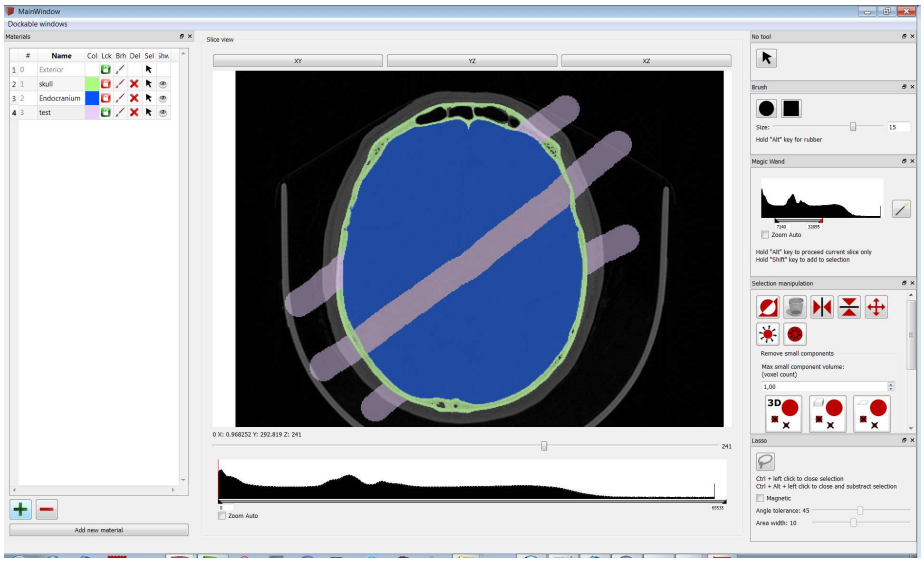

Figure 34: Material lock scenario 3: selection crossing skull (locked) and endocranium (locked) added to the "test" material

16.3 Tools

16.3.1 No tool

Click on the arrow button to unselect the last used tool. This is usefull to "disable" the left click in the slice view.

16.3.2 Brush

Select the brush shape by hitting one of both buttons, and set up the size with the slider.
On the slice view, the mouse will appear as a blue brush.
Note that the brush is drawn with the real Image 3D pixel size. With non-isometrics voxels, on orthoslices, the brush is drawn with rectangular pixels (see fig 35).
If you move the mouse, the brush will move only on the Image 3D pixels. So, if you zoom a lot, the brush will be displayed aliased and will moove jerky.
Left click and hold left button to draw. The blue brush will disappear during movement to let you see the image 3D parent behind.
Hold the "Alt" key to transfrom the brush to a rubber, and erase the selection (see 36).

16.3.3 Magic wand

From a clicked voxel, the magic wand will grow while the grey level of neighbor is inside the range chosen on the histogram.
To use the magic wand, you should first select a range of grey values in the histogram, click on the magic wand button, and then, click a voxel in the slice view (see fig 37).
The growth is computed in 3D. Hold the "Alt" key to proceed with the current slice only (see fig 38).

If a selection is already set, it will be erased, and replaced by the magic wand computation. To keep the selection, and add the magic wand computation, hold the "Shift" key.
You can hold "Alt" and "Shift" keys to use the magic wand many times on the current slice.
After a voxel click in the slice view, you can change the grey range on the histogram. If you do so, there are 2 scenarios:

- The last magic wand computation was in 3D: the computation will be updated only when you release the histogram range slider.
- The last magic wand computation was on current slice only: the computation will be updated in real time while moving the histogram range slider.

This is due to the computation time needed by the 3D growth. A real time update would have slow dramatically TIVMI.

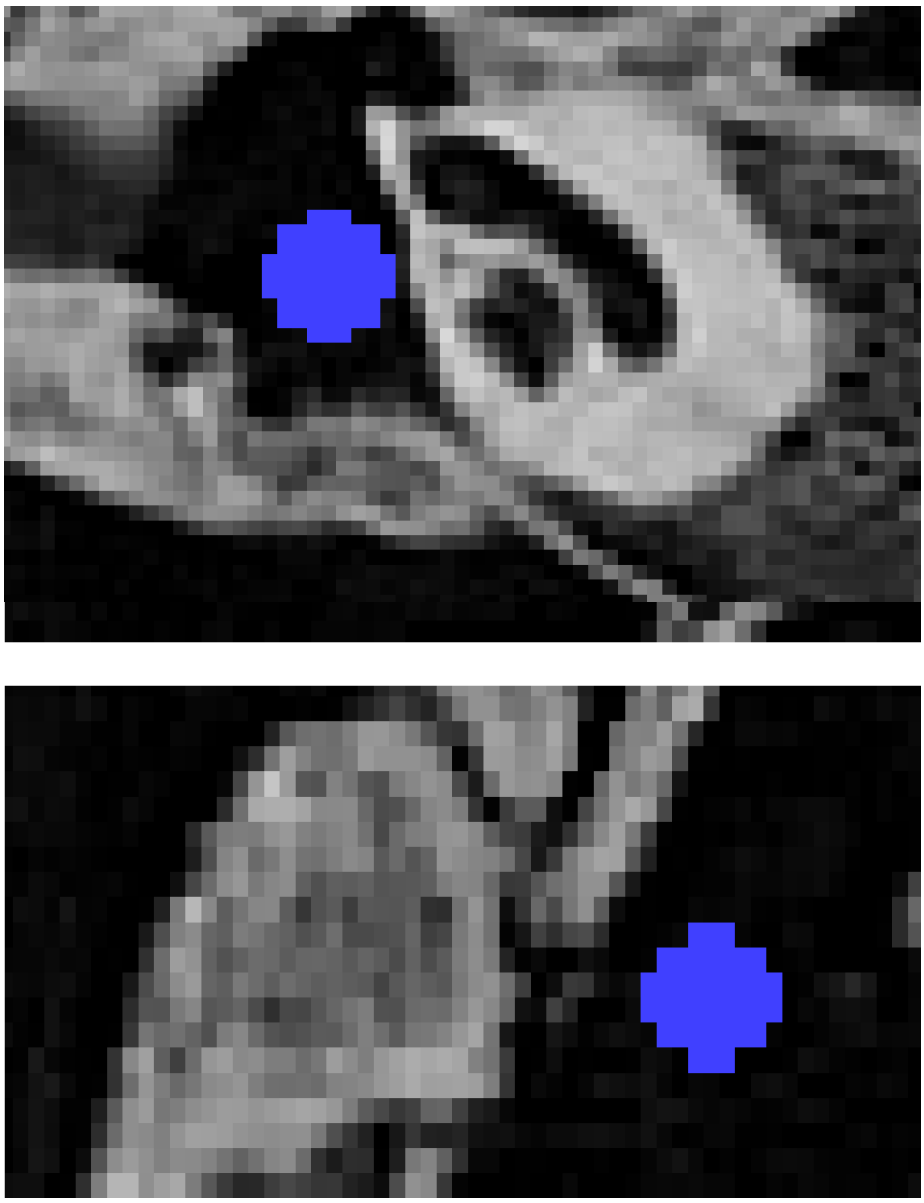

Figure 35: Segmentation: brush tool shown in blue with square pixels (top) and rectangular pixels (bottom) regarding slice orientation on non-isometrics voxels.

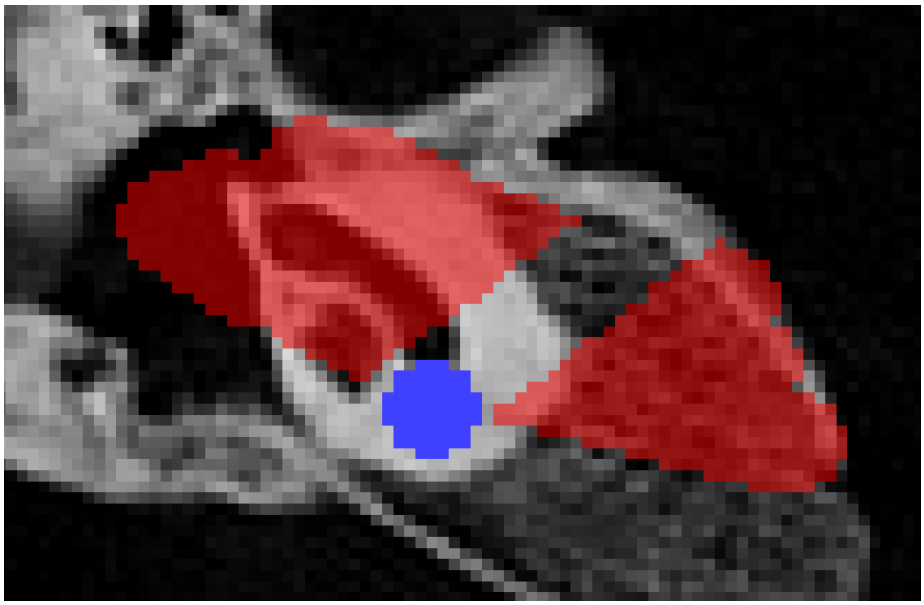

Figure 36: Segmentation: brush tool draws or erase a selection.

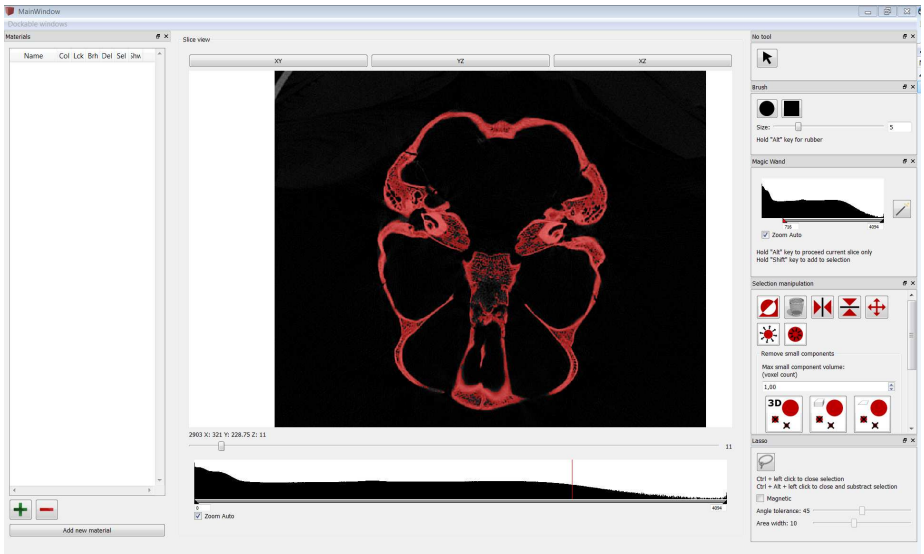

Figure 37: Segmentation: magic wand tool in 3D. The selection grows and stops when the voxel grey level is outside the range chosen in the histogram. The propagation is in 3D.

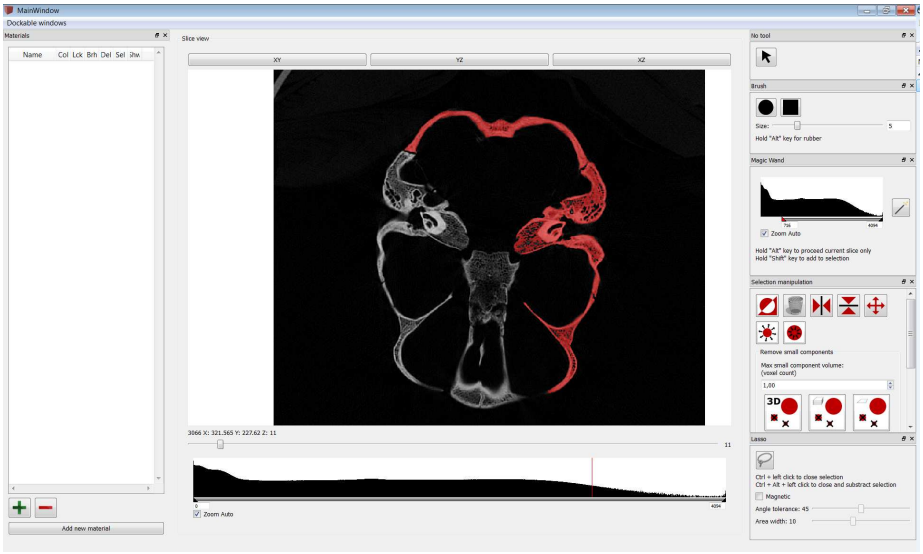

Figure 38: Segmentation: magic wand tool on current slice. The selection grows and stops when the voxel grey level is outside the range chosen in the histogram.

16.3.4 Selection manipulation

- Invert: invert selection on the 3D Image. Hold “Alt” key to invert current slice only (see fig 39).
- Clear: sets all voxels as unselected. Hold “Alt” key to clear current slice only.
- Horizontal mirror: makes the symmetry of the selection along a vertical axis. On XY view or XZ view, this will make a X mirror. On YZ view, this will make a Y mirror (see fig 40).
- Vertical mirror: makes the symmetry of the selection along an horizontal axis. On YZ view or XZ view, this will make a Z mirror. On XY view, this will make a Y mirror.
- Move: moves the selection in 3D (see fig 41). If a part of the selection is out of the image dimensions, it will be kept during the move, but lost when the mouse button is released.
- Grow: grows the selection in 3D. Each unselected voxel next to a selected voxel (in 6 connectivity) will be selected (see fig 42).
- Shrink: shrinks the selection in 3D. Each selected voxel next to an unselected voxel (in 6 connectivity) will be unselected.

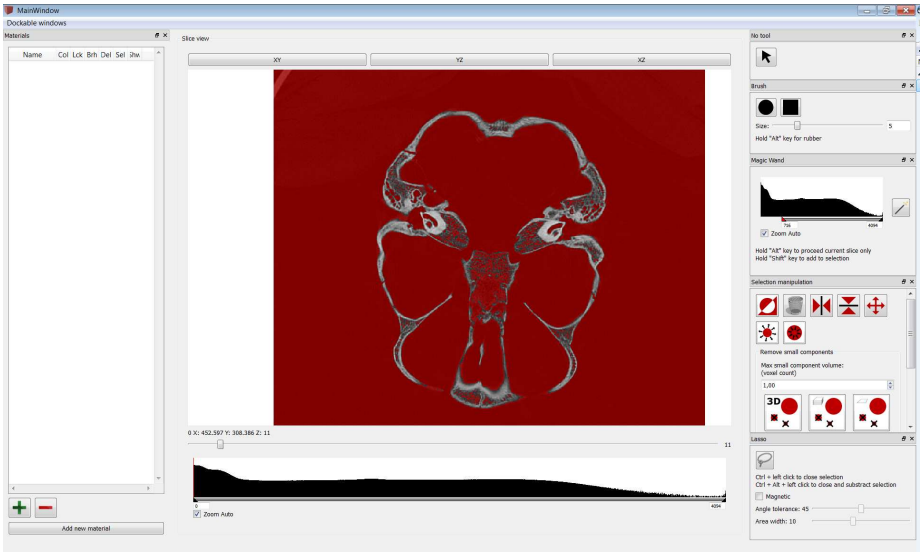

Figure 39: Segmentation: inverted selection: all is selected but the bone previously selected.

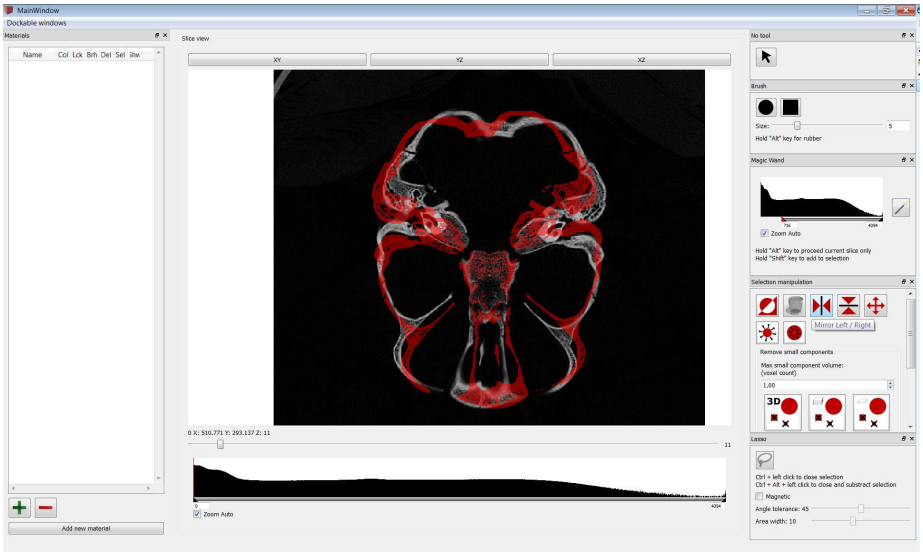

Figure 40: Segmentation: horizontal mirror of a selection.

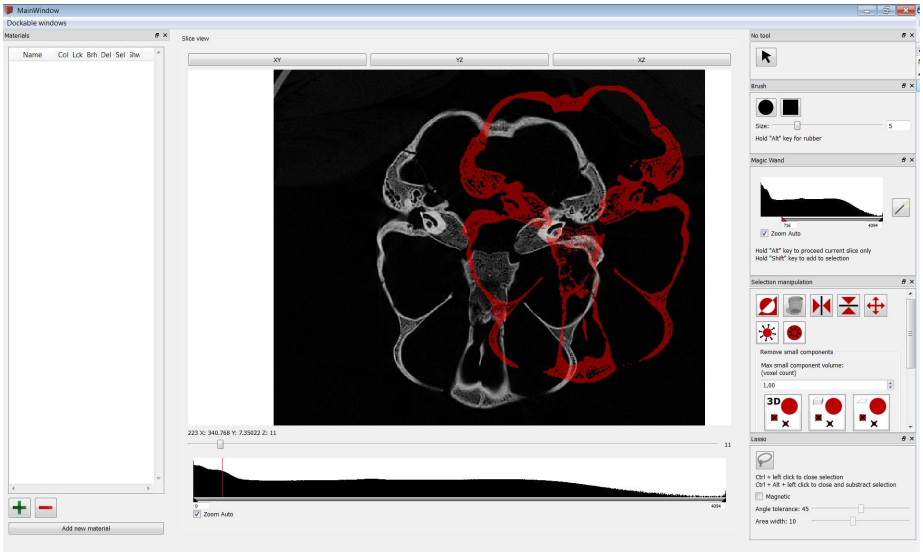

Figure 41: Segmentation: move selection.

- Remove small components: looks for small sets of selected voxels, and unselect them if count is lower or equal to the max small component volume (see fig 43). This can be done in 3D, on current slice, or on all slices reagrding the selected slice orientation.

- Fill holes: looks for small sets of unselected voxels closed inside selected voxels, and select them if count is lower or equal to the max hole volume or if all holes are targeted (see fig 44). This can be done in 3D, on current slice, or on all slices reagrding the selected slice orientation.

16.3.5 Lasso

Click on the lasso button to select this tool.
On the slice view, hold left button to make a free hand drawing.
Release left button and move the mouse. A yellow line shows the next segment in real time. Left click to validate this line (see fig 45).
To close the lasso, and fill interior, hold the "Ctrl" key and left click anywhere. This click will not be a part of the lasso.
When closing the lasso, you can hold the "Alt" key to substract lasso to selection rather than adding it.
Magnetic: the magnetic option, customisable with the 2 sliders "Angle tolerance" and "Area width", will use the parent image to make local HMH computation in 3D, and attract the lasso on shapes. When you move the mouse, the lasso drawn in red will be validated by the next click, while the yellow segment helps you to find the good shape to follow.

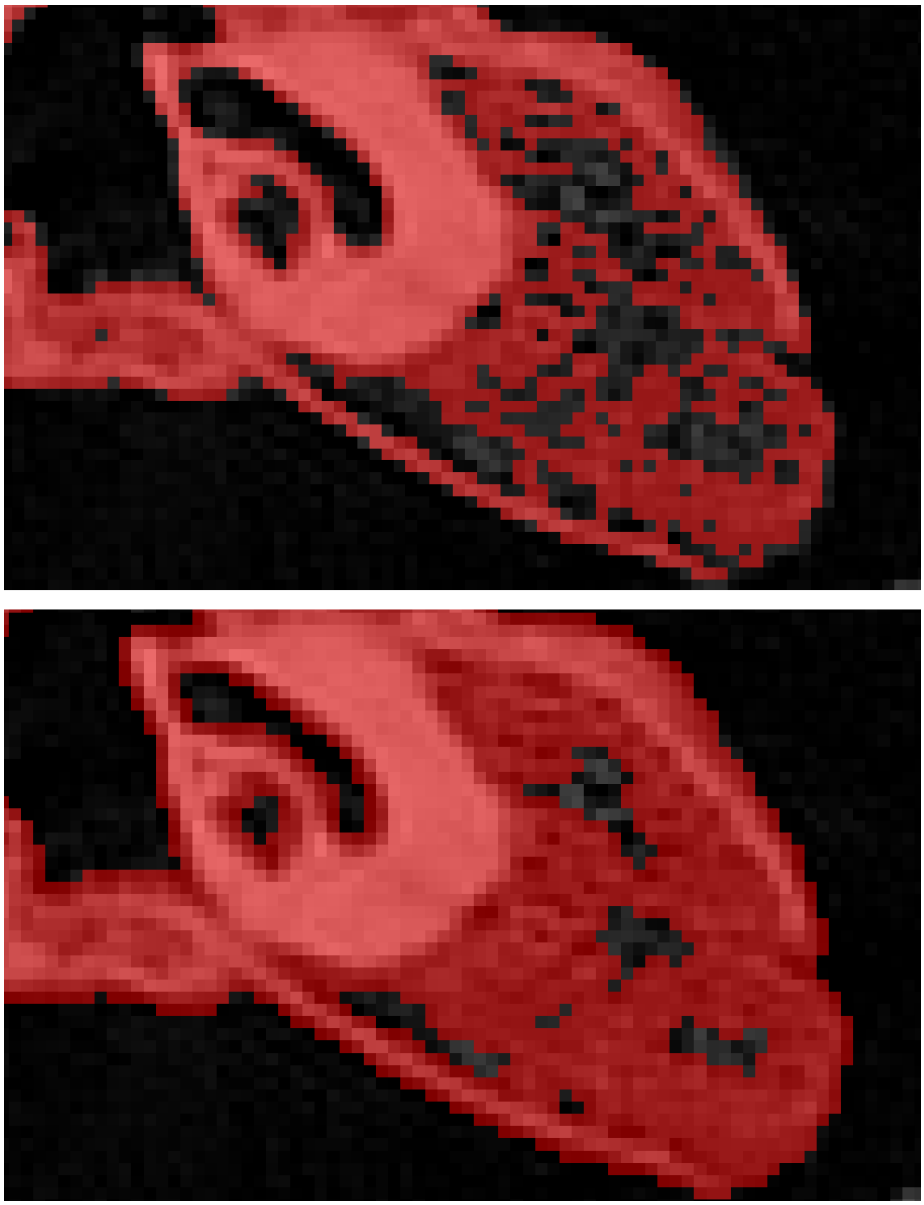

Figure 42: Segmentation: grow selection (top before growing, and bottom after growing.

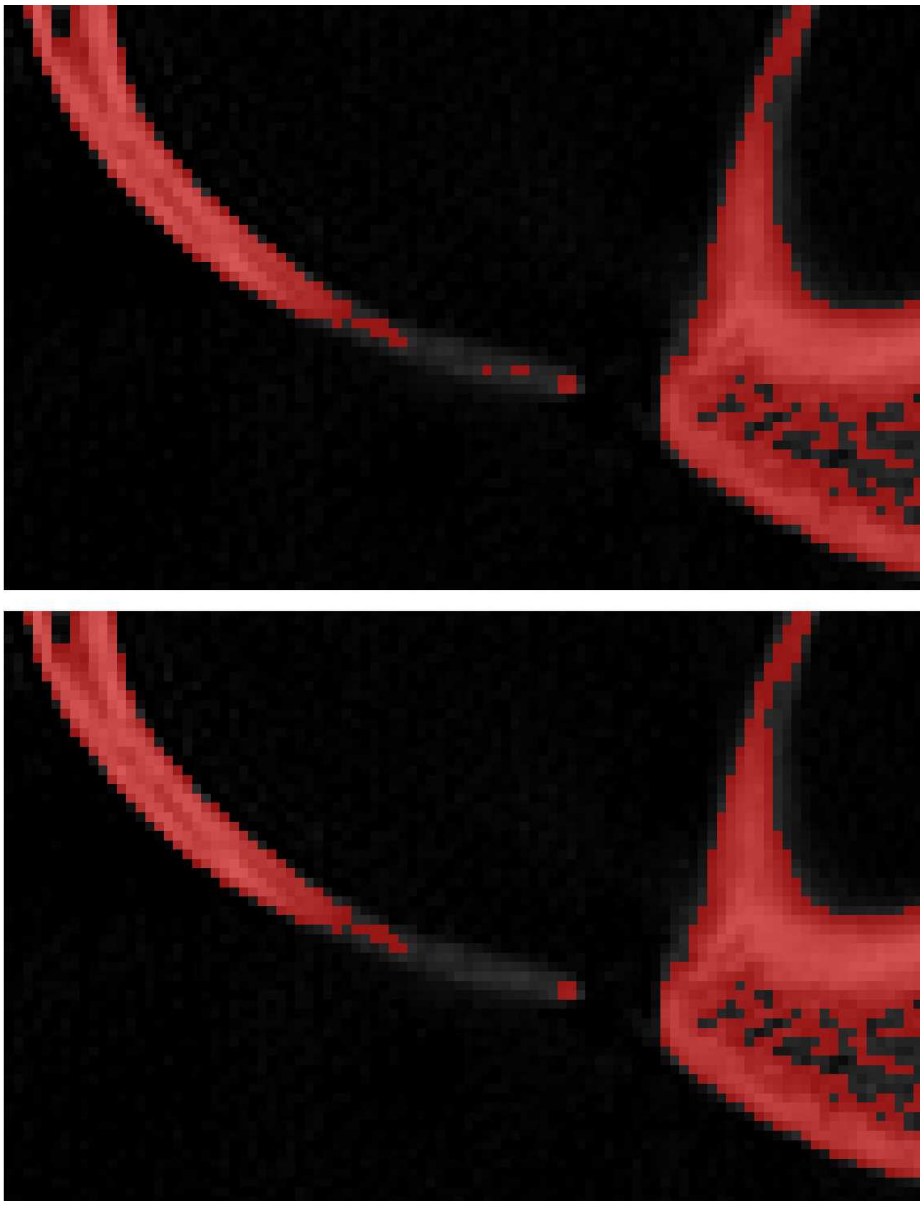

Figure 43: Segmentation: removes small components. On the top, the original selection, and on the bottom, the small components which volume was lower or equal to 2 voxels are removed.

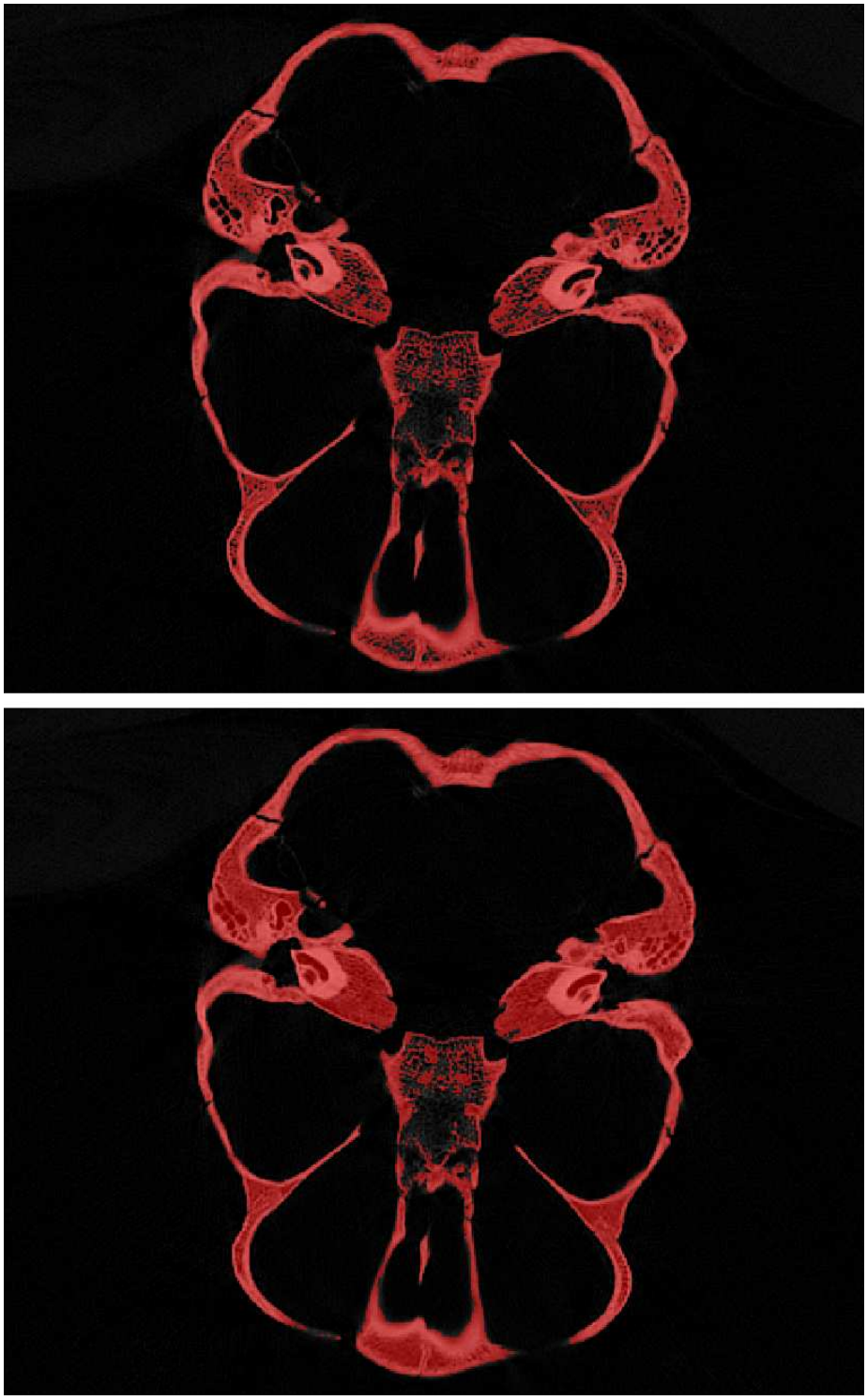

Figure 44: Segmentation: fill holes. On the top, the original selection, and on the bottom, all holes have been filled.

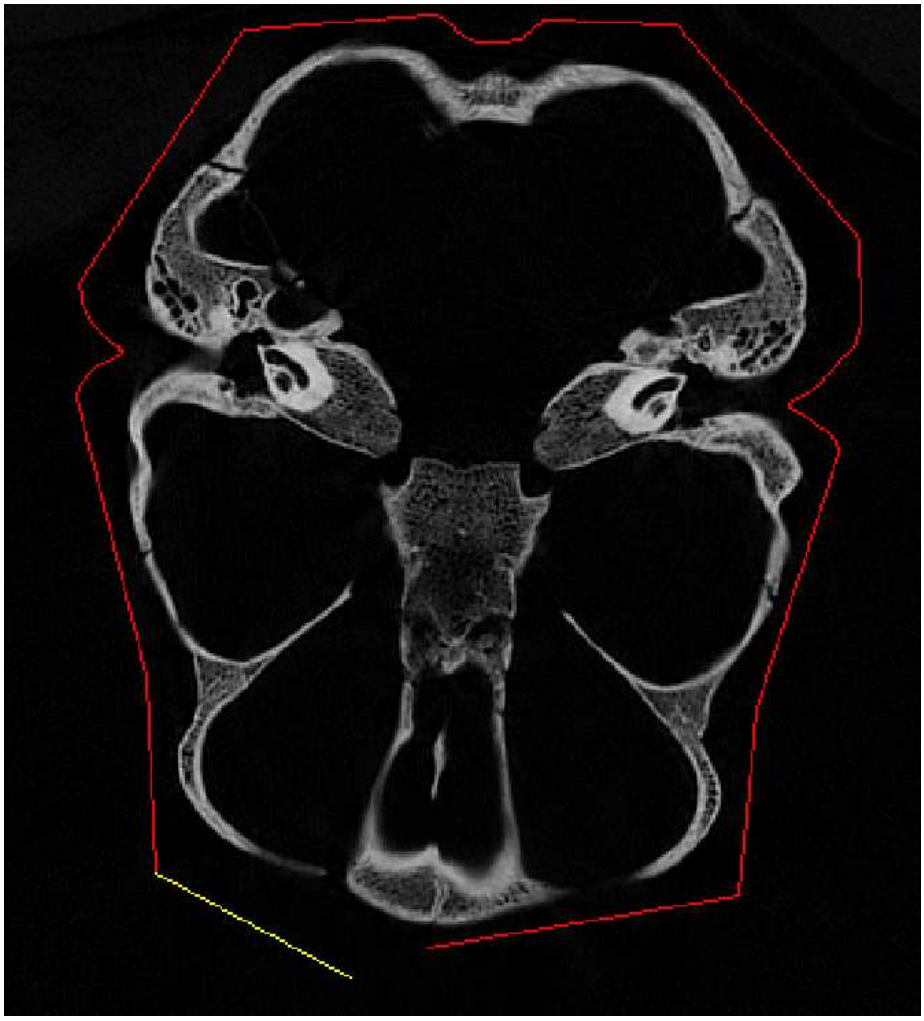

Figure 45: Segmentation: lasso. Parts of drawing are made by free hand, and others by lines.

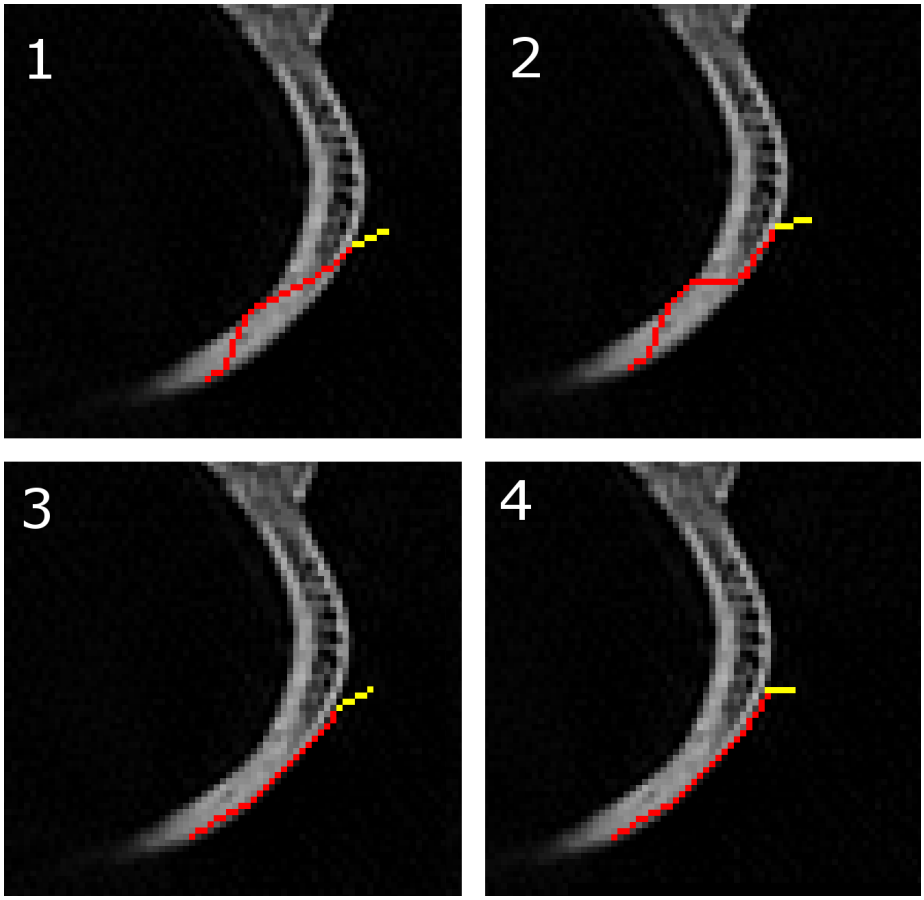

Figure 46: Segmentation: Magnetic lasso. 1: Angle=45ï¿½ and Area 10. 2: Angle=10ï¿½ and Area 10. 3: Angle=10ï¿½ and Area 5. 4: Angle=45ï¿½ and Area 5.

The "Angle tolerance" in degrees determines how the lasso will stuck to the shapes. The "area width" is the area where the HMH is computed. Big values will detect greater contrasts while small values will follow light shapes (see fig 46).

17 What can I do with a segmentation?

After editing a segmentation, you will get a 8-bits depth 3D image where grey level 0 (zero) is not a region, and others values are different regions.
From this 3D image, you can use every plugins which use a 8-bits 3D image as input.
The most probable use should be the surface extraction of a region to get a 3D

mesh. This can be done by using the HMH surface plugin (see 5.5). This plugin can use a 8-bits 3D image as a binary image to extract the surface. The HMH Surface plugin does not support segmented images with materials, but it is able to extract the surface of non-zero voxels set in the 8-bits image. So, if you made a segmentation with only one material, this material will be reconstructed as a 3D mesh. If your segmentation contains more materials, they all will be merged into one.
To extract only one material, you must use the plugin double threshold (see 4.3) on the segmented image and set the label number of the chosen material as grey value 1 and 2, and create a new 8-bits 3D image. Then, use this new image as the binary image needed by the HMH surface plugin. More information are avalilabale in the scenarios 25.
It is planned to update plugins in order to support segmentations.

Part VI
Scripts

18 Scripting language

The scripting language is basic and allows to reproduce all actions of the interface.
The best way to learn scripting language is to record yourself while using TIVMI, and edit the script file with an ASCII text editor.

18.1 Commands on objects

Always begin with `CMD` followed by the name of object to target, followed by a command keyword, and arguments if available. Arguments are int for integer numbers, float for floating point numbers, and string for an object name and diplayed between `<` and `>`.

18.1.1 Image 3D

- `CMD <Image3D> setCurrentSlice <int num>`: to set the displayed slice to "num".
  e.g. `CMD scan setCurrentSlice 3`
- `CMD <Image3D> setVoxelDimensions <float width> <float height> <float thickness>`: to change voxel dimension.
  e.g. `CMD scan setVoxelDimensions 0.25 0.25 0.5`
- `CMD <Image3D> updateSlicesPositionsFromVoxelResolution <float spaceBetweenSlices>`: to update slices position. In fact, the voxel thickness is not necessarily the space between slices. Use this command to setup new location of slices.
  eg: `CMD scan updateSlicesPositionsFromVoxelResolution 0.25`
- `CMD <Image3D> mirrorX`: to invert x positions of voxels.
- `CMD <Image3D> mirrorY`: to invert y positions of voxels.
- `CMD <Image3D> ShowBoundingBox`: to show the bounding box of the 3D Image.
- `CMD <Image3D> HideBoundingBox`: to hide the bounding box of the 3D Image.

18.1.2 Point 3D

- `CMD <Point3D> setColor <int red> <int green> <int blue>`: to set the 3D point color. red, green and blue are 3 integers from 0 to 255.
  e.g. for orange: `CMD P1 setColor 255 127 0`
- `CMD <Point3D> showName <true|false>`: to show of hide the nam of point in the 3D view.
  e.g. `CMD P1 showName false`
- `CMD <Point3D> setSize <float size>`: to set the point size.
  e.g. `CMD P1 setSize 1.2`
- `CMD <Point3D> setStyle <int style>`: 0 for z-slice, 1 for ball.
  e.g. `CMD P1 setStyle 1`
- `CMD <Point3D> setPosition <float x> <float y> <float z>`: to set the position in the 3D space;
  e.g. `CMD P1 setPosition 0.0 10.0 -2.5`
- `CMD <Point3D> setLocalPosition <float x> <float y> <float z> <string reference>`: same as setPosition but using the basis of object "reference".
  e.g. set postion of P1 to the voxel (10,15,3) of the 3D image scan: `CMD P1 setPosition 10 15 3 scan`

18.1.3 Plane

- `CMD <Plane>setConstraint <string object>`: to set limitations to object to display the plane. Object can be an image3D, a mesh, a boundingbox, or a path3D.
  e.g. `CMD frontal setConstraint head`
- `CMD <Plane>showNormal <0|1>`: 1 to show normals, 0 to hide the normal.
  e.g. `CMD francfort showNormal 1`
- `CMD <Plane>setColor <int red> <int green> <int blue>`: to set the plane color. red, green and blue are 3 integers from 0 to 255.
  e.g. for purple: `CMD frontal setColor 127 95 127`
- `CMD <Plane>invertNormal`: to invert the normal.
  e.g. `CMD frontal invertNormal`
- `CMD <Plane>shift <float distance>`: to shift the plane along its normal. If distance is negative, the plane will be shifted backward the normal.
  e.g. `CMD frontal shift 42.67`

18.1.4 Mesh

- `CMD <Mesh> setColor <int red> <int green> <int blue>`: to set the mesh color.
  e.g. for bone color: `CMD head setColor 235 235 220`
- `CMD <Mesh> setShininess <0|1>`: 1 to enable the shininess, 0 to disable it.
  e.g. `CMD head setShininess 1`
- `CMD <Mesh> buildNormals <0|1>`: 0 is normal per face, 1 is normal per vertex.
  e.g. `CMD head buildNormals 1`
- `CMD <Mesh> scale <float factor>`: to scale the mesh.
  e.g. `CMD head scale 10`
- `CMD <Mesh> optimizeVertexUse`: to delete duplicates vertices.
  e.g. `CMD head optimizeVertexUse`
- `CMD <Mesh> showNormals <true|false>`: to show or hide normals.
  e.g. `CMD head showNormals true`
- `CMD <Mesh> invertNormals`: to invert normals.
  e.g. `CMD head invertNormals`
- `CMD <Mesh> setPointSize <float size>`: to set the size of vertices, only for point cloud.
  e.g. `CMD timeOfLightScan setPointSize 3`
- `CMD <Mesh> setResolution <float resolution>`: to set the display resolution of point cloud. Resolution is a percentage from 0 excluded to 100 included.
  e.g. `CMD timeOfLightScan setResolution 50`
- `CMD <Mesh> setLDResolution <float resolution>`: to set the low definition display resolution of point cloud. The low definition resolution is a percentage from 0 excluded to 100 included used when moving the camera in the 3D view.
  e.g. `CMD timeOfLightScan setLDResolution 10`
- `CMD <Mesh> setColorPerVertex <0|1>`: 1 to use the color per vertex (if available), 0 to use the general mesh color.
  e.g. `CMD timeOfLightScan setColorPerVertex 1`

18.1.5 Bounding Box

- `CMD <BoundingBox> setColor <int red> <int green> <int blue>`: to set the bounding box color.
  e.g. for red: `CMD roi1 setColor 255 0 0`

- `CMD <BoundingBox> setCorner1 <float x> <float y> <float z>`: to set the coordinates of corner 1.
  e.g. `CMD roi1 setCorner1 25.34 62.56 -12.45`

- `CMD <BoundingBox> setCorner2 <float x> <float y> <float z>`: to set the coordinates of corner 2.
  e.g. `CMD roi1 setCorner2 -23.92 -20.41 72.12`

18.1.6 Segment 3D

- `CMD <Segment3D> setColor <int red> <int green> <int blue>`: to set the segment color.
  e.g. for dark green: `CMD headWitdh setColor 32 64 32`

- `CMD <Segment3D> setSize <float size>`: to set the segment size (balls).
  e.g. `CMD headWitdh setSize 4.5`

- `CMD <Segment3D> setPoint1Position <float x> <float y> <float z>`: to set the position of point 1.
  e.g. `CMD headWitdh setPoint1Position 28.10 38.61 -30.62`

- `CMD <Segment3D> setPoint2Position <float x> <float y> <float z>`: to set the position of point 2.
  e.g. `CMD headWitdh setPoint1Position -39.27 61.95 -50.51`

18.1.7 Straight line

- `CMD <StraightLine> setConstraint <string object>`: to set limitations to an object for the plane display. Object can be an image3D, a mesh, a boundingbox, or a path3D.
  e.g. `CMD rotationAxis setConstraint head`

- `CMD <StraightLine> setColor <int red> <int green> <int blue>`: to set the line color. red, green and blue are 3 integers from 0 to 255.
  e.g. for pink: `CMD rotationAxis setColor 255 127 127`

- `CMD <StraightLine> setSize <float size>`: to set the line size.
  e.g. `CMD rotationAxis setSize 3`

18.1.8 Path 3D

- `CMD <Path3D> setColor <int red> <int green> <int blue>`: to set the path color. Red, green and blue are 3 integers from 0 to 255.
  e.g. for blue: `CMD perimeter setColor 0 0 255`

- `CMD <Path3D> showPointNumber <true|false>`: to swho or hide point number in the 3D view.
  e.g. `CMD perimeter showPointNumber true`

- `CMD <Path3D> setSize <float size>`: to set the path size (balls).
  e.g. `CMD perimeter setSize 2`
- `CMD <Path3D> setStyle <0|1>`: 0 to display points as balls, or 1 to display points as squares on screen.
  e.g. `CMD perimeter setStyle 1`
- `CMD <Path3D> setClosed <true|false>`: false to keep the path open, true to close it.
  e.g. `CMD perimeter setClosed true`
- `CMD <Path3D> addPointAfter <int position> <float x> <float y> <float z>`: to add a point which coordinates are (x,y,z) after the point position. Position of point in a 3D path starts from 1.
  e.g. `CMD perimeter addPointAfter 5 3.6 1.8 -5.2`
- `CMD <Path3D> addPointBefore <int position> <float x> <float y> <float z>`: to add a point which coordinates are (x,y,z) before the point position. Position of point in a 3D path starts from 1.
  e.g. `CMD perimeter addPointBefore 1 -2.3 4.6 -1.8`
- `CMD <Path3D> deletePoint <int position>`: delete the point at the specified position.
  e.g. `CMD perimeter deletePoint 2`
- `CMD <Path3D> movePoint <int position> <float x> <float y> <float z>`: to move the point at the specified position to the specified coordinates (x,y,z). Position of point in a 3D path starts from 1.
  e.g. `CMD perimeter movePoint 1 2.0 -4.9 -7.8`

18.1.9 AFA3D

- `CMD <Afa3D> setSkullVisible <true|false>`: to show or hide the skull mesh.
  e.g.: `CMD specimen setSkullVisible true`
- `CMD <Afa3D> setSkullPointsVisible <true|false>`: to show or hide 3D points of the skull mesh.
  e.g.: `CMD specimen setSkullPointsVisible true`
- `CMD <Afa3D> setTargetedPointsVisible <true|false>`: to show or hide 3D points of the targeted face.
  e.g.: `CMD specimen setTargetedPointsVisible true`
- `CMD <Afa3D> setComputedFaceVisible <true|false>`: to show or hide the computed face mesh.
  e.g.: `CMD specimen setComputedFaceVisible true`

- `CMD <Afa3D> setComputedPointsVisible <true|false>`: to show or hide 3D points of the computed face.
  e.g.: `CMD specimen setComputedPointsVisible true`
- `CMD <Afa3D> setOriginalMeshColor <int red> <int green> <int blue>`: to set the skull mesh color. Red, green and blue are 3 integers from 0 to 255.
  e.g. for white: `CMD specimen setOriginalMeshColor 255 255 255`
- `CMD <Afa3D> setSkullPointsColor <int red> <int green> <int blue>`: to set the 3D points color of the skull. Red, green and blue are 3 integers from 0 to 255.
  e.g. for red: `CMD specimen setSkullPointsColor 255 0 0`
- `CMD <Afa3D> setTargetedPoints1Color <int red> <int green> <int blue>`: to set the class 1 3D points color of the targeted face. Red, green and blue are 3 integers from 0 to 255.
  e.g. for dark red: `CMD specimen setTargetedPoints1Color 127 0 0`
- `CMD <Afa3D> setTargetedPoints2Color <int red> <int green> <int blue>`: to set the class 2 3D points color of the targeted face. Red, green and blue are 3 integers from 0 to 255.
  e.g. for dark green: `CMD specimen setTargetedPoints1Color 0 127 0`
- `CMD <Afa3D> setComputedFaceColor <int red> <int green> <int blue>`: to set the computed face color. Red, green and blue are 3 integers from 0 to 255.
  e.g. for skin: `CMD specimen setComputedFaceColor 255 244 231`
- `CMD <Afa3D> setComputedPoints1Color <int red> <int green> <int blue>`: to set the class 1 3D points color of the computed face. Red, green and blue are 3 integers from 0 to 255.
  e.g. for red: `CMD specimen setComputedPoints1Color 255 0 0`
- `CMD <Afa3D> setComputedPoints2Color <int red> <int green> <int blue>`: to set the class 2 3D points color of the computed face. Red, green and blue are 3 integers from 0 to 255.
  e.g. for green: `CMD specimen setComputedPoints2Color 0 255 0`
- `CMD <Afa3D> movePoint <int index> <float x> <float y> <float z>`: to set the indexth point position. index is an integer between 0 and 77. x, y and z are floating point numbers.
  e.g. for the Nasion `CMD specimen movePoint 4 10.5671 -25.4713 5.8417`

18.1.10 Group

- `CMD <Group> removeFromGroup [<string objectName1>] [<string objectName2>]`: to remove "objectName1", "objectName2" etc from the group named "Group".
  e.g. `CMD study removeFromGroup P1 P2`

- `CMD <Group> addToGroup [<string objectName1>] [<string objectName2>]`: to add "objectName1", "objectName2" etc to the group named "Group".
  e.g. `CMD study addToGroup P1 P2`

18.2 Commands to create objects

Always begin with `NEW` followed by the type of object, followed by the name of the new object to create, followed by arguments if available. Arguments are int for integer numbers, float for floating point numbers, and string for a sentence and displayed between `<` and `>`.

- `NEW Point3D <string name> <float x> <float y> <float z>`: to create a 3D point at coordinates (x,y,z).
  e.g. `NEW Point3D origin 0.0 0.0 0.0`
- `NEW Segment3D <string name> <float p1x> <float p1y> <float p1z> <float p2x> <float p2y>` to create a 3D segment from point (x1,y1,z1) to (x2,y2,z2).
  e.g. `NEW Segment3D totalLength -1.0 0.0 3.5 8.6 0.0 -5.2`
- `NEW BoundingBox <string name>`: to create a new bounding box.
  e.g. `NEW BoundingBox my_roi`
- `NEW Path3D <string name> <float x> <float y> <float z>`: to create a 3D path containing 1 point at coordinates (x,y,z).
  e.g. `NEW Path3D perimeter 8.5 -3.7 2.9`
- `NEW GenericInt <string name> <int value>`: to create an integer number.
  e.g. `NEW GenericInt examCount 3`
- `NEW GenericFloat <string name> <float value>`: to create a floating point number.
  e.g. `NEW GenericInt supposedVolume 563.98`
- `NEW GenericString <string name> ["]<string value>["]`: to create a sentence. Double brackets are optional.
  e.g. `NEW GenericString readMeFirst "Look at the front side of this bone"`
- `NEW Afa3D <string name> [<string originalMeshName>]`: to create an Afa3D object. If provided, the Afa3D object will be based on a mesh.
  e.g. `NEW Afa3D specimen aSkullMesh`
- `NEW Group <string name> [<string objectName1>] [<string objectName2>] ...` : to create a group named "name" containing "objectName1", "objectName2" etc.
  e.g. `NEW Group study myMesh aPoint1 aPoint2 aSegment`

18.3 Regular expressions

Always begin with `RE` followed by the string containing commands and regular expression.
e.g. `RE Show #^point[0-9]+$#`

18.4 Delete objects

Always begin with `DELETE` followed by the name of object to delete.
e.g. `DELETE P1`

18.5 Show objects

Always begin with `SHOW` followed by the name of object to show.
e.g. `SHOW P1`

18.6 Hide objects

Always begin with `HIDE` followed by the name of object to hide.
e.g. `HIDE P1`

18.7 Rename objects

Always begin with `RENAME` followed by the name of object to rename, and followed by the new name.
e.g. `RENAME P1 P2`

18.8 Call a plug-in

Always begin with `CALLPLUGIN` followed by the name of the plugin (see chapter IV), and followed by input list and output list.
Input and output lists are lists of arguments separated with space.
Each argument of input list is prefixed by `in:`
Each argument of output list is prefixed by `out:`
After the `:`, use object names, numbers or strings.
e.g. `CALLPLUGIN Image3D_DoubleThreshold_Image3D in:scan in:10534 in:38610 out:binaryScan`

19 Script generation

The "Script" menu contains the "Start record" command that begins the generation of the script. User must use specific object names to make them recognizable for further use of the script in other TIVMI states (i.e. saves).

Part VII
Scenarios

20 Open a DICOM exam, and place 3D points on slices

1. Menu plugin, Image3D, load dicom
2. Input: browse a folder containing a list of dicom files
3. Output: give a name: scan for example
4. If the load dicom plugin detects multiple series, it will create more than one image3D and number them.
5. Click on OK
6. Click on "show / hide", image 3D, scan
7. In the "camera and environement" subwindow, the top combolist will contain "scan".
8. Click on "set target"
9. The image3D scan will appear in the 3D view. Use the mouse (right button down) to unzoom the 3D view if the scan is not visible
10. In the "Image 3D" subwindow, move the scroll bar slice to navigate into slices.
11. In the "Image 3D" subwindow, move the scroll bar window width and window center to change the contrats.
12. Use the mouse (left button down) in the 3D view to rotate around the image 3D.
13. In the "Point 3D" subwindow (Figure 47), click on the "pick new" button in the "3D picking group" (don't use the "HMH 3D picking" group).
14. Click on the wanted structure in the 3D view to create the point. The location of the landmark must then be visualized prior to clicking on "pick new".
15. Give a name: p1. Each landmark should be given a specific and unique name if scripts are to be computed.
16. In the "Point 3D" subwindow, it is possible to change the style to "ball" and change the size and color.

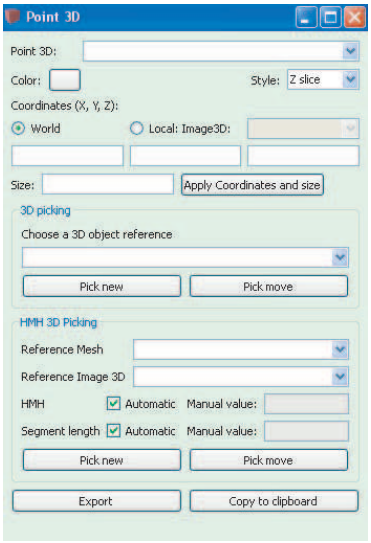

Figure 47: Point 3D subwindow.

21 Create a segment and find the HMH interface between two tissues

There are two ways to create segments. Use a plugin to create a segment based on two existing landmarks or create directly a segment.

21.1 Create a new segment

1. Localize the position of the first landmark beginning the future segment on the slice (or surface) in the 3D view
2. From the "Segment 3D" subwindow (Figure 48), click on "Pick new" and click on the target landmark location.
3. Give a name: seg1
4. The next click on the 3D view will place the second landmark of the segment and create the segment.
5. If one of the point is not well positioned, select the seg1 segment in the combo list and click on "Pick move point 1" (or 2) to change the location of the landmark and length of the segment.
6. The coordinates of the 2 landmarks along with the length of the segment can be exported or copied to the clipboard from the "Segment 3D" subwindow

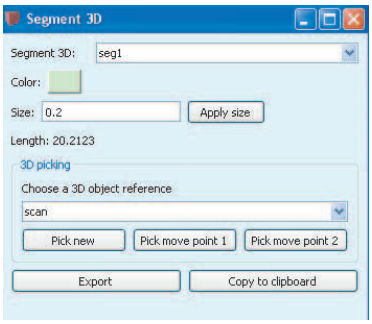

Figure 48: Segment 3D subwindow.

21.2 Create a segment from 2 existing 3D points

1. Go to: menu plugin, segment 3D, make a 3D segment from 2 points
2. Input: In the first combo list, choose the first landmark.
3. In the second combo list, chose the second landmark.
4. Output: give a name: seg2
5. Click on "show / hide", segment 3D, segp1p2. In the "Segment 3D" subwindow, the length of the segment is specified (the unit is the one by default in the DICOM file, usually mm).

21.3 Evaluate the density profile on a segment

The HMH profile can be observed on a segment in TIVMI. This operation is indicative.

1. Menu Plug-in, Plot, Plot profile on 3D segment.
2. Select the target Image 3D and the Segment 3D on which the profile will be measured.
3. Name the plot to create.
4. In the "Plot" subwindow (Figure 49), select the plot created and show it.
5. The first point of the segment corresponds to the beginning of the plot (the end being point 2). Tha variation in density can be observed through soft tissues and bone (Figure 50).
6. The interface between 2 tissues can be manually calculated by selecting two extreme densities on the plot and by clicking on "HMH". The Y value expresses the density in Hounsfield units.

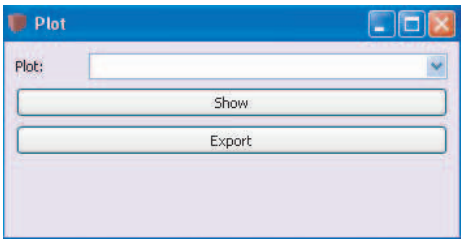

Figure 49: Plot subwindow.

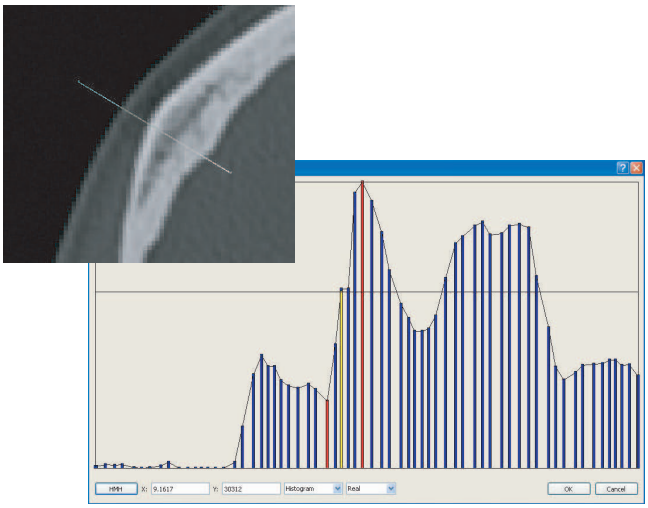

Figure 50: HMH profile led on a segment passing through the cranial vault.

21.4 Automatically compute the interface 3D points on a 3D segment with the HMH protocol

The previous operation can be automatically performed and the interface 3D points placed with reproducibility on a segment.

1. Plug-in, Point3D, Point3Ds HMH Segments3D HMH Value
2. Select the Image and Segment 3D on which the HMH will be calculated. Enter a HMH value (0 for automatic HMH calcuation is advised. If there are several tissues and the user want to target a specific one, he cen enter the corresponding HMH value previously evaluated in a plot).
3. Finally, name the prefix of the points to be created.
4. The points are created, make them appear in Show / Hide, Point3D (Figure 51).

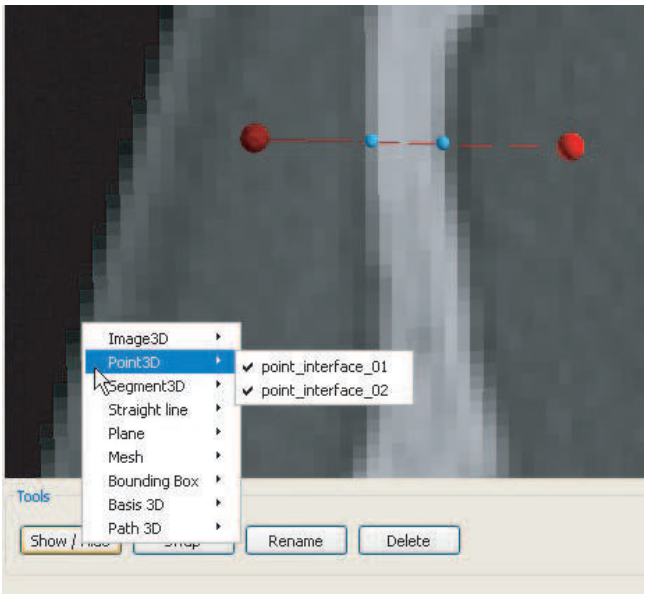

Figure 51: Automatic placement of interface landmarks (blue) on a segment (defined by the red landmarks).

22 Reconstruct two surfaces from two different densities

Craniometrics imply landmarks that cannot be spotted on 2D slices and reproducible surface reconstruction based on the HMH protocol can be done from medical CT scan. This scenario develop the different steps to reconstruct hard and soft tissues of the face from a medical exam.

22.1 Select the region of interest

Depending on the virtual memory available, surface reconstruction may be time consuming and it is not always necessary to render the whole exam when only a region is studied. In this cases, the user can define a box around the interested zone and reconstruct only that zone. Also, this is faster than the slice by slice segmentation process.

1. Once the exam is loaded and displayed in the 3D view, select the slice where the first corner of the bounding box will be set.
2. Create the bounding box from the specific subwindow (Figure 52)
3. Once the new box is named, place the first corner by clicking on "Pick Corner 1" then clicking on the target zone in the 3D view.

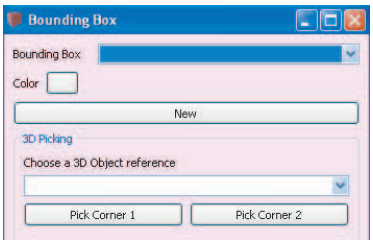

Figure 52: Bounding Box subwindow.

4. Scroll the exam up to the slice where the opposite corner of the box will be set

5. Place the second corner by clicking on "Pick Corner 2" in the Bounding Box subwindow. The bos is created (Figure 53)

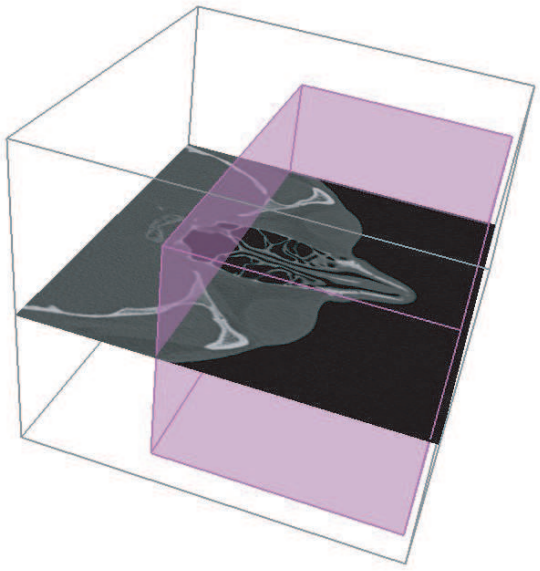

Figure 53: Bounding box created.

6. Change the color of the created box if the visibility is not good and change slices to check if all the region of interest is into the bounding box. The two opposite corners can be moved by repeating the operation "Pick Corner" to adjust the box.

7. A new scan can be cropped from the scan according to the bounding box. Reconstruction of surfaces from this new scan will be faster. Use Plug-in, Image3D, Crope a 3D Image, select the image, the box, and name the new scan.

22.2 Reconstruct HMH surfaces

In the case of medical exam and presence of tissues of different densities, we can manually find an optimal threshold for each material.

1. In the "Image 3D" subwindow, with a Window width set at 0, the user looks for specific values in Hounsfield units until only the wanted structures are visible (Figure 54). This subjective phase of choosing threshold will not interfere with the reproducibility of the reconstruction as the algorithm uses the HMH protocol.

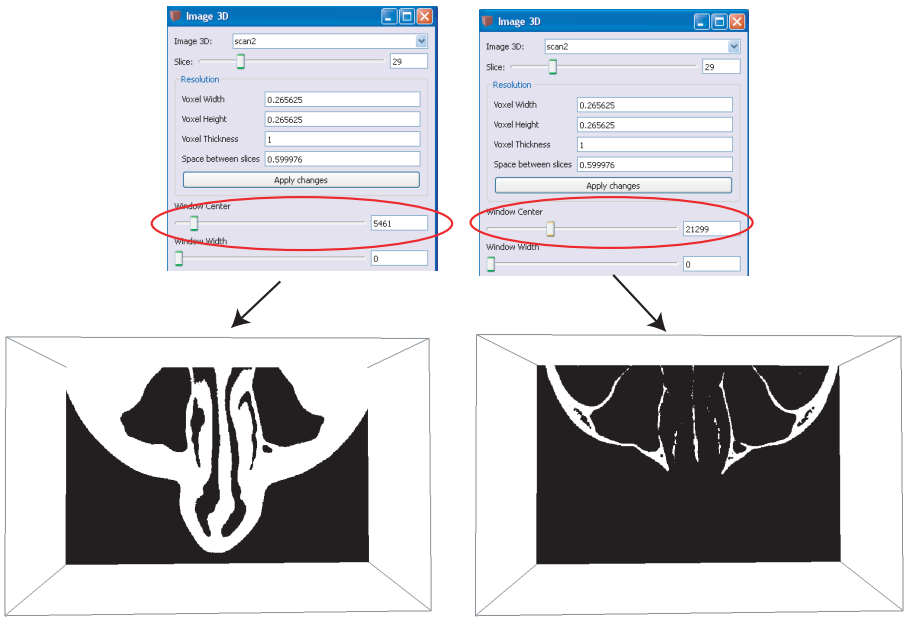

Figure 54: Manual visualization of threshold unit in case of different tissues.

2. The HMH reconstruction can be performed through Plug-in, Mesh, HMH Surface. Specify the Image3D that will be used to calculate the HMH interfaces in 3D.

3. For medical exam, we advise to use the free mode (0).

4. Gray threshold is the value previously recorded to aim a particular density.

5. Gray min and max can be used to avoid the recosntruction of material of exterme density (artifacts). By default, enter respectively 0 and 65535.

6. HMH max length must be at least equivalent to twice a voxel diagonal in order to cover at least 2 voxels in each direction when measuring the HMH (for a voxel of 0.3 x 0.3 x 1, the 3D diagonal averages 1.08 so the segment should be superior to 2.16, putting 3 is a wise choice).

7. The Binary image is to be used only if such an image has been previously created, in this example it is not used.

8. Finally, name the mesh to create (Figure 55). Calculation of the surface can be more or less time-consuming depending on the DICOM resolution and the exam size.

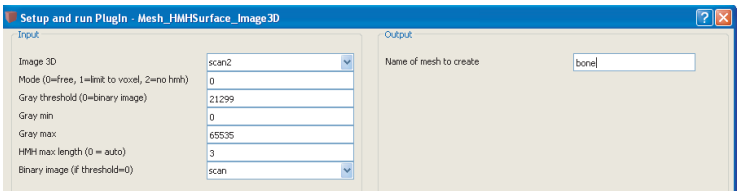

Figure 55: Setup of the HMH Surface Plug-in.

9. Once the reconstruction is done, TIVMI indicate the potential errors and memory amount allocated for the operations (Figure 56).

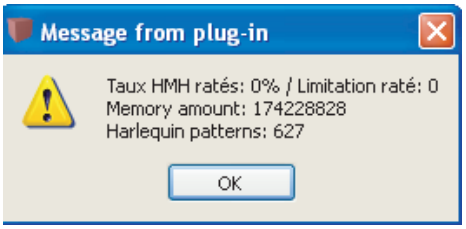

Figure 56: Automatic message after a successful HMH surface reconstruction.

10. Use the Show / Hide toolbox to show the mesh created.

11. The color mesh can be changed in the Mesh subwindow (Figure 57) and the light can also be modified for better visualization in the Camera and Environment subwindow.

The same operation can be performed by changing the threshold to obtain the surface of the skin. Furthermore, in the Mesh subwindow, the last surface displayed can be made transparent by changing its opacity (Figure 58). If any problem, hide all meshes and show them in order (the last being the first visible layer).

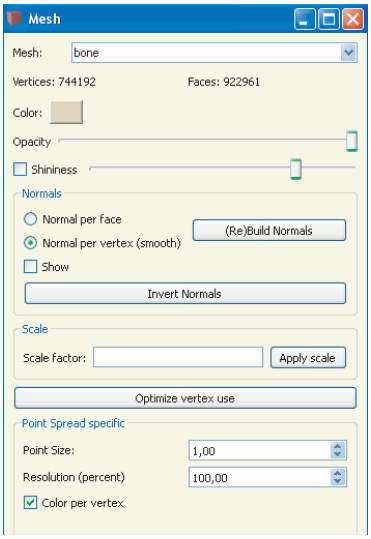

Figure 57: Mesh subwindow.

23 Enhanced craniometrics

Anthropologic measurements can me performed directly on the reconstructions. Most landmark definitions need the skull to be oriented according to the Frankfurt plane, this section present the construction of planes and the positioning of some traditional landmarks. Further examples of specific measurements are also described.

23.1 Plane construction and landmark positioning

Geometric landmarks are defined as "the most superior" or "the most anterior" point of structures. This orientation is given by the Frankfurt Horizontal, which definition is that of a plane passing through both porion (most superior point of the external acoustic meatus) and both infraorbitale (most inferior point on the orbital rim). As a plane passes through three landmarks, we advise to create a medium plane based on those four points for a better accuracy.

1. Using the 3D picking in the Point 3D subwindow, place the right and left porion and orbitale on the reconstructed skull (Figure 59). The nasion (at the metting of the frontal and the internasal suture) can also be placed for the further construction of the sagittal plane.

2. The creation of the planes can be scripted (Script menu, Start record). Create four planes passing by both porions and left infraorbitale, both porions and right infraorbitale, both infraorbitale and right porion, both infraorbitale and left porion. Use the Plug-in, Plane, Make plane and

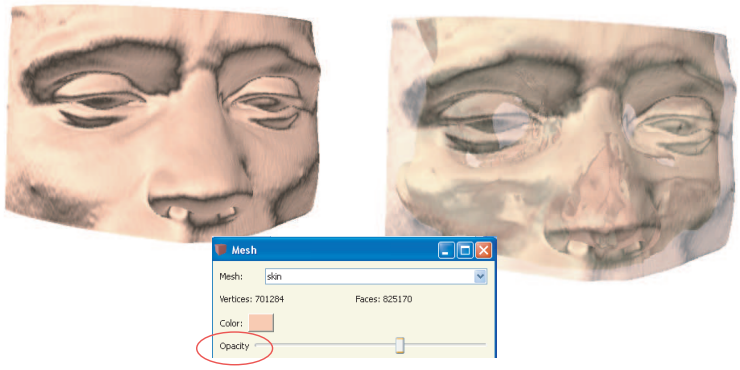

Figure 58: Transparency of mesh.

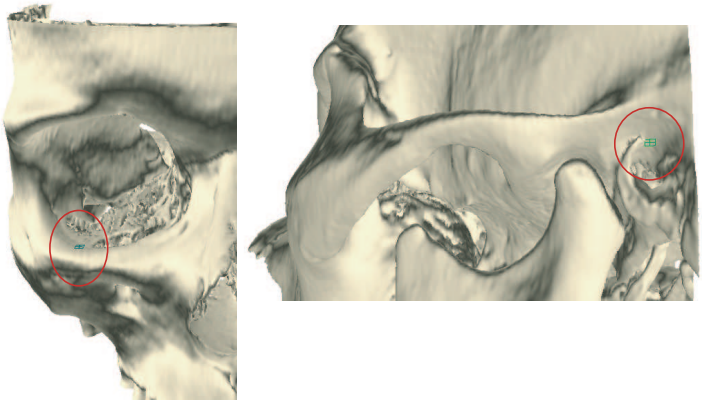

Figure 59: Placement of the infraorbitale and porion.

name them FH1 to FH4. As objects, the planes must be displayed (Show / Hide, Plane) and applied to a specific constraint (image 3D or mesh) through the Plane subwindow (Figure 60).

3. Each pair of plane can thus be used to create a mean plane (Plug-in, Plane, Medium plane) and the two resulting mean planes can be averaged together to get the final Mean Transverse (MT) plane. When using medium planes, the user must be careful to orient the normal of each plane in the same direction. The normal of the planes can be displayed and inverted in the Plane subwindow (Figure 60).

4. The frontal or coronal plane can be constructed by creating a plane orthogonal to MT and passing through both porion, another one passing through both infraorbitale (Plug-in, Plane, Make an orthogonal plane thru

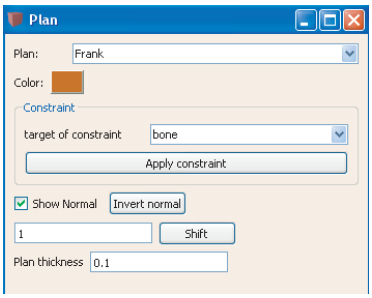

Figure 60: Plane Subwindow.

2 points). The Mean Coronal (MC) plane can finally be averaged between those two.

5. The third plane, the Mean Sagittal, can be constructed simply by creating a plane orthogonally to MT and MC passing through the nasion (Plug-in, Plane, Orthogonal Plane from 2 planes through 1 point)(Figure 61).

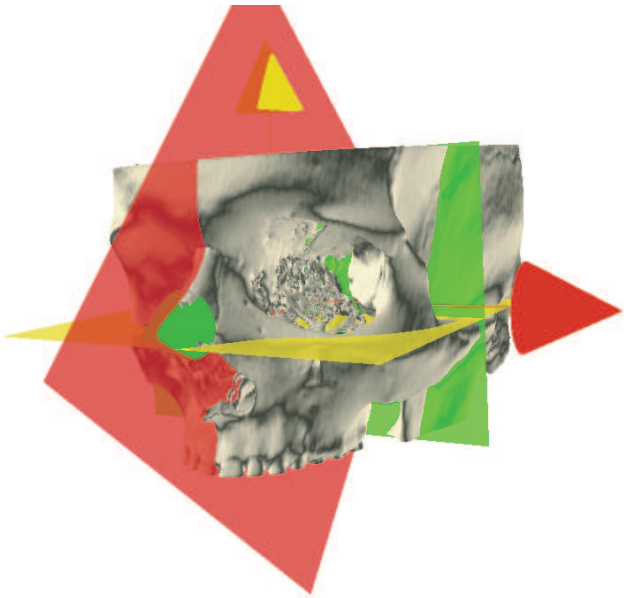

Figure 61: Mean planes (yellow = mean transverse ; green = mean coronal ; red = mean sagittal) with normal displayed.

6. Color, thickness and position of the planes can be modified in the Plane subwindow.

7. For the positioning of geometric landmarks as the glabella (most anterior point of the cranial vault between the supraorbitary arches), the MC can be shifted or thickenned until delimiting the targeted zone (Figure 62).

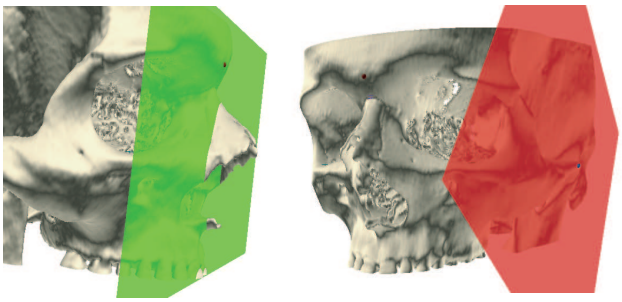

Figure 62: Plane-assisted positioning of the glabella with the MC and zygion with the MS.

8. This methodology can be adapted to all geometrically defined traditional landmarks. It significantly enhances the repeatability and reproducibility of anthropometric protocols.

23.2 Measurement in projection

TIVMI proposes basic geometric operations that can be used to perform specific measurements. Mastoid height or orbital height in example, are measurements that are not strictly defined by two landmarks. As those were created to be taken with a sliding caliper, it requires more than one step to record them.

1. For mastoid height (MDH), the most inferior point of the mastoid process (mastoidale) can be positioned by thickenning the MT up to its very position. In this case, it is important to keep the original position of the MT as MDH is the direct measurement from the tip of the mastoid process to the Frankfurt Horizontal.

2. Start recording a script here. The mastoidale can thus be orthogonally projected on the MT with Plug-in, Point3D, Orthogonal Projection: Point On Plane. Select the mastoidale and the MT and name the projected mastoidale.

3. Create a segment between the mastoidale and the projected mastoidale (Plug-in, Segment3D, Make 3D segment from 2 points and name it MDH). The projected mastoidale will not be used anymore, it can be deleted. Stop recording the script here. MDH can be consulted in the Segment subwindow (Figure 48).

Orthogonal projections are also used to record the orbital height (OBH) which is the projected measurement (vertically) between the superior and in-

ferior orbital rims. The most superior (suprakonchion) and most inferior (infraorbitale) landmarks of the orbital rims are, however, not aligned vertically, a point to point (direct) measurement will not give the traditional OBH as if it was recorded with a sliding caliper.

1. Position the most inferior and superior landmarks of the orbital rim (visual positionning is sufficient for small structures as orbits).
2. In order to record a straight projected measurement of OBH, it is necessary to project the landmarks both on MS and MC. Start recording a script now if you want to further automatise those operations.
3. Project the landmarks on the MS (Plug-in, Point 3D, Orthogonal Projection: Point on Plane). Project those projected landmarks on the MC (Figure 63).

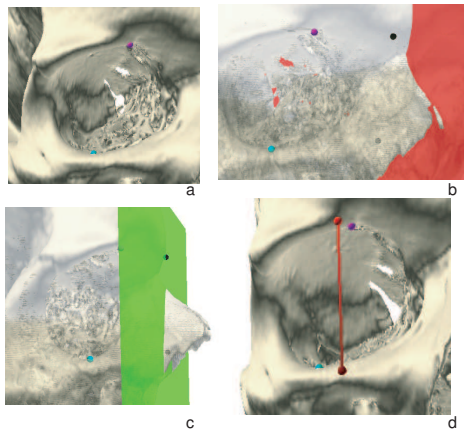

Figure 63: (a) Most superior and inferior landmarks of the orbital rim. (b) Projection of both landmarks on the MS. (c) Projection of the projected landmarks on the MC. (d) Shifted segment equivalent to the traditional measurement OBH.

4. Make a 3D segment from the last projected landmarks and name it OBH (Plug-in, Segment3D, Make 3D segment from 2 points). Delete all projected landmarks. Stop recording the script here. OBH length can be consulted in the 3D segment subwindow.
5. For a better visibility, the 3D segment can be shifted (Plug-in, Segment 3D, Segment 3D Shift Segment 3D Along Plane Normal with ratio). It creates another segment, the original one can be deleted.

23.3 Measurement of soft tissue depth

Since both hard and soft tissues can be reconstructed with medical CT scans, the soft tissue depth (STD) can be measured at specific landmarks. To enhance

the repeatability and reproducibility of such measurements, projection and automatic determination of the interface between skin and air can be computed in TIVMI.

1. The glabella has been positioned using the MC plane. Start a script here.
2. Use the projection of point following the normal of a plane (Plug-in, Point 3D, Shift point along the normal of a plane). In this case, we want to measure the STD at the glabella, we will project it according to the MC (Figure 64).

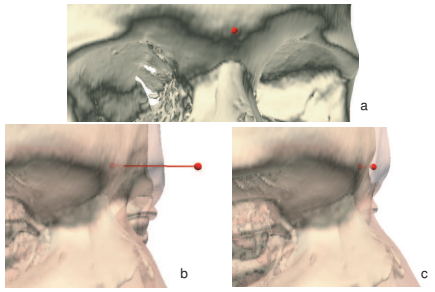

Figure 64: (a) Bony glabella positioned. (b) Segment created from the projected bony glabella along the normal of the MC plane. (c) STD at the glabella between the bony glabella and the projected interface between skin and air calculated by HMH.

3. First get the point "out" of the bone, in order to avoid the bone density to bias the further HMH calculation. Use a ratio of 0.5 (units are in mm as DICOM are most of the time in mm) and name the point glabella1.
4. Then repeat the same operation but make sure that the second point crosses the skin. Use a ratio of 30 and name the point glabella2.
5. Create a 3D segment from glabella1 and glabella2 and name it glabseg.
6. Let TIVMI find automatically the HMH interface between skin and air by performing the calculation on the glabseg segment (Plug-in, Point 3D, Point3D HMH Segment 3D HMH Value).Let 0 for automatic HMH and name the interface glabellasoft.
7. Make appear the created interface point (Show / Hide, Point 3D, glabella soft interface 01). Delete the glabella1, glabella2 and glabseg, they won't be used anymore.
8. Create a segment between the bony glabella and the soft glabella, name it glabellaSTD. You can also erase now the interface point. Stop recording the script here. The STD can be checked in the segment subwindow.

24 Segmentation of a 3D image: basic threshold

1. Open a 3D image, and show it in the 3D view
2. In the object control windows, hit on the "Edit a new segmentation" button (see fig 65)

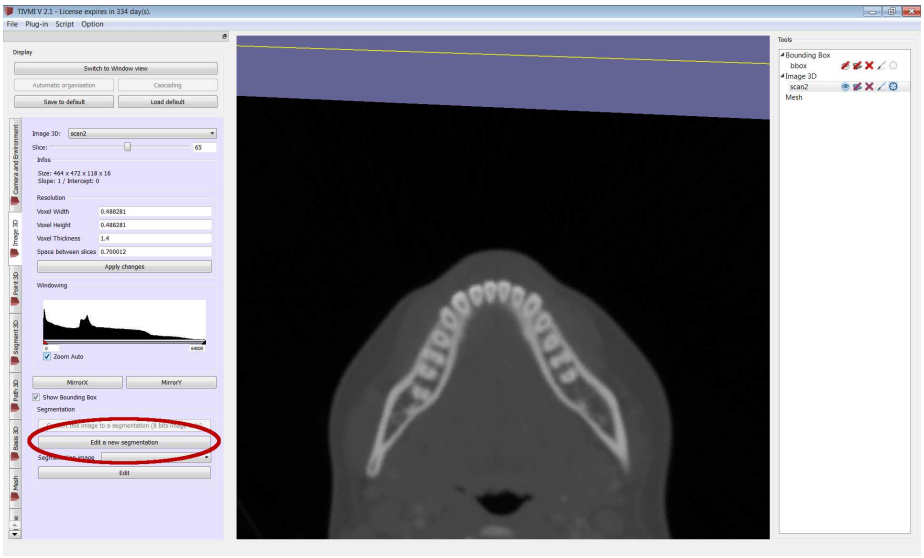

Figure 65: Open a 3D image and create a new segmentation.

3. Create a new material by hitting the "add new material" button (see fig 66)
4. Rename the material: double click on the name.
5. Choose a color: click on the color square. We suggest you to choose a color different from a grey levels (used to display CT scans), and red (used to display current drawings) (see fig 67)
6. Use the magic wand to select a bone (see fig 68 and 69:)
   (a) Select a range on the histogram to keep only high values of grey level
   (b) Click on the magic wand icon
   (c) Hold "alt" key and click on a bone in the image to proceed on the current slice only
   (d) Move the histogram range to adjust the threshold. Computation is in real time because the last click was on current slice only
   (e) Click again on a bone in the image to proceed for all slices

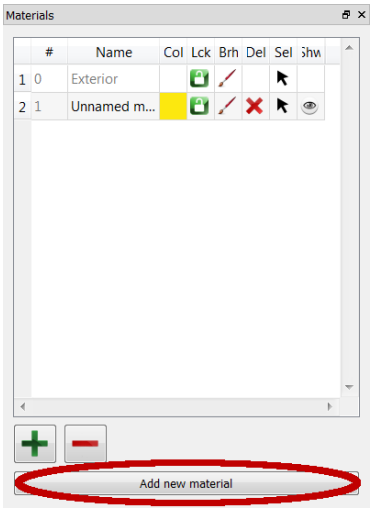

Figure 66: Add a new material.

(f) You can move again the histogram range to adjust the threshold, but the result is updated only when you release the range because the last click was on all slices

7. Click on the bone material in the material list (see fig 70)
8. Click on the green "+" button to add the selection to the "bone" material (see fig 70)
9. Close the segmentation window
10. You can see a new 3D image in the object list named "seg_" followed by the name of the original 3D image see fig 71
11. Create a surface on the segmented image using the HMH surface plugin (menu plug-in → mesh → HMH Surface 2). Setup entries as follow:
    (a) Image 3D: the original 3D image (the CT-scan)
    (b) Mode: 0
    (c) Gray threshold: 0 (because you will use the segmented image to define the object to extract
    (d) Gray min: 0
    (e) Gray max: 65535 (for a 16 bits CT-scan) see 5.5
    (f) HMH max length: 0
    (g) Binary image: choose the "seg_..." 3D image previoulsy created

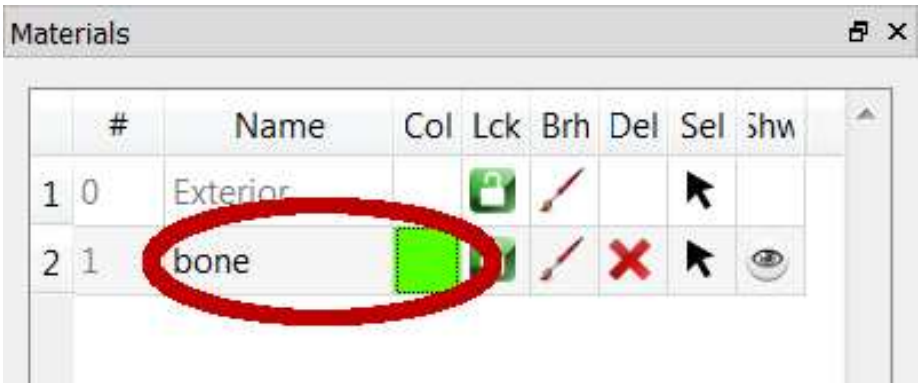

Figure 67: Rename and choose a color for the new material.

The surface extraction works on non zero voxels of a binary image. The segmented image containing only one material, as this scenario creates, is considered as a binary image. If you want to create 2 or more materials and extract surfaces of those materials, see the scenario 25.

25 Advanced segmentation of a 3D image

We assume that you already know basic segmentation as described in the scenario 24.
We start from the scenario 24 where a mandible and a maxillary are in a material named "bone". We will try to create a material for the mandible and another for the maxillary.

1. Create a new material named "maxillary"
2. Rename the "bone" material to "mandible" matreial
3. Set the "exterior" material to drawing disabled
4. Choose a slice orientation to see the fusion between the mandible and the maxillary (sagittal view for instance)
5. On each slice, use the magnetic lasso to draw a line between the mandible and the maxillary. We recommand to set the "area width" and "angle tolerance" to a small value in order to help the HMH computation to find the interface near the yellow line of the lasso (see fig 72)
6. Hold the left mouse button to roughly select the maxillary, then "ctrl" click to close it (see fig 73 and fig 74). It doesn't matter if you don't select all the maxillary on the slice, especially if you zoom a lot so the maxillary is partially hidden, we will correct this latter. Note that the

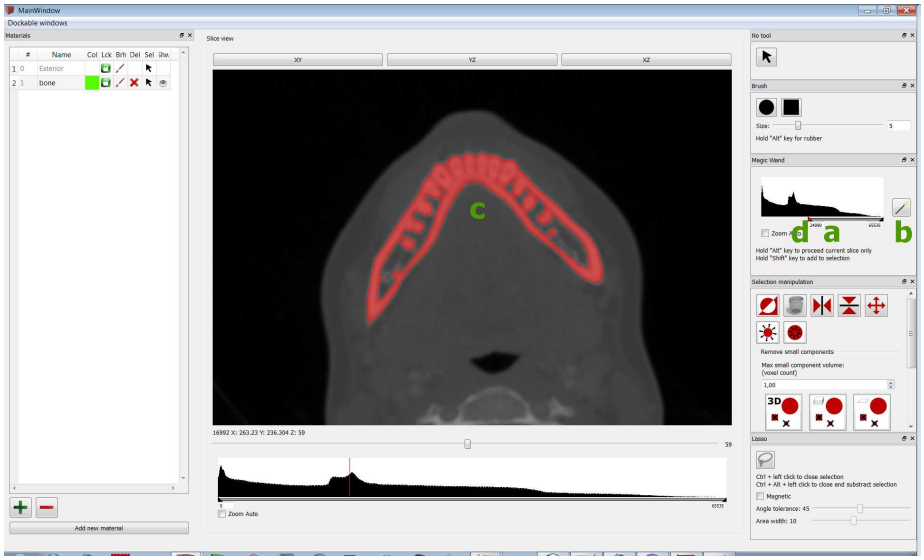

Figure 68: Magic wand usage on current slice only.

exterior material is set to drawing disabled, so the selection in red is not set on the exterior.

7. Add the selection to the maxillary material (see fig 75)

8. Now, we need to complete the maxillary with remaining parts owned by the mandible material: use the brush and paint roughly on those remaining parts on each slice. Note that the brush will not affect voxels owned by the exterior material (see fig 76).

9. Add selection to the maxillary material.

Now, there are 3 materials: 0: exterior, 1: mandible, and 2: maxillary.
You can close the segmentation editor.
To extract the surface of the mandible and the maxillary, we will use the HMH surface plugin. But this plugin uses a binary image. Our segmented image is not a binary image because it contains values 0, 1 and 2. If you use the segmented image as a binary image for the HMH surface extraction, the plugin will extract the surface of non-zero voxel values. So it will construct the surface of the mandible and the maxillary merged.
In order to have 2 distinct meshes, we need to call the HMH surface on each material:

1. Call the plugin image3D → double threshold

2. Set the input image to your segmented image

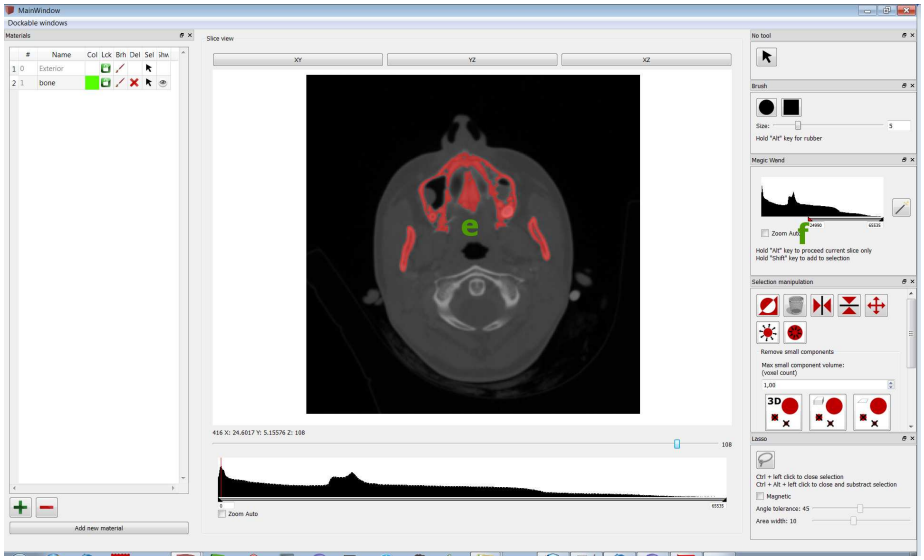

Figure 69: Magic wand usage in 3D.

3. Set Gray value 1 and gray value 2 to 1: it means that all voxels which value is 1 will be set to 255, and all others voxels will be set to 0. The binary image will contain only the mandible

4. Set a name for this binary image. e.g. mandible_binary

5. Call again the plugin image3D → double threshold

6. Set the input image to your segmented image

7. Set Gray value 1 and gray value 2 to 2: The binary image will contain only the maxillary

8. Set a name for this binary image. e.g. maxillary_binary

9. Create a surface on each binary image using the HMH surface plugin (menu plug-in → mesh → HMH Surface 2). Setup entries as follow:

   (a) Image 3D: the original 3D image (the CT-scan)

   (b) Mode: 0

   (c) Gray threshold: 0 (because you will use a binary image to define the object to extract

   (d) Gray min: 0

   (e) Gray max: 65535 (for a 16 bits CT-scan) see 5.5

   (f) HMH max length: 0

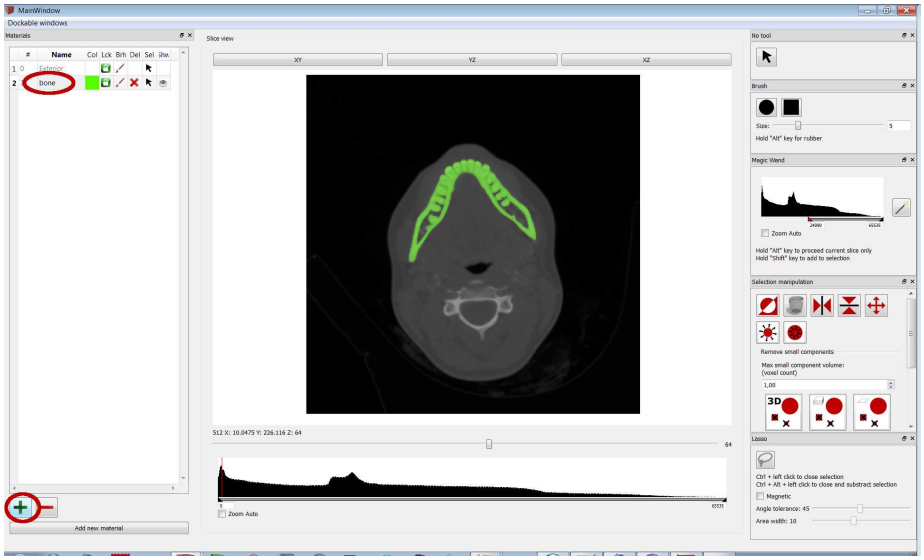

Figure 70: Add a selection to a material.

(g) Binary image: choose the "mandible_binary" 3D image previoulsy created (and "maxillary_binary" for the second call)

10. You can delete both binary images

11. Show both meshes and choose a color (see fig 77)

The HMH surface plugin will evolve to create automatically a mesh by material. The binary image in input will be replaced by a segmented image.

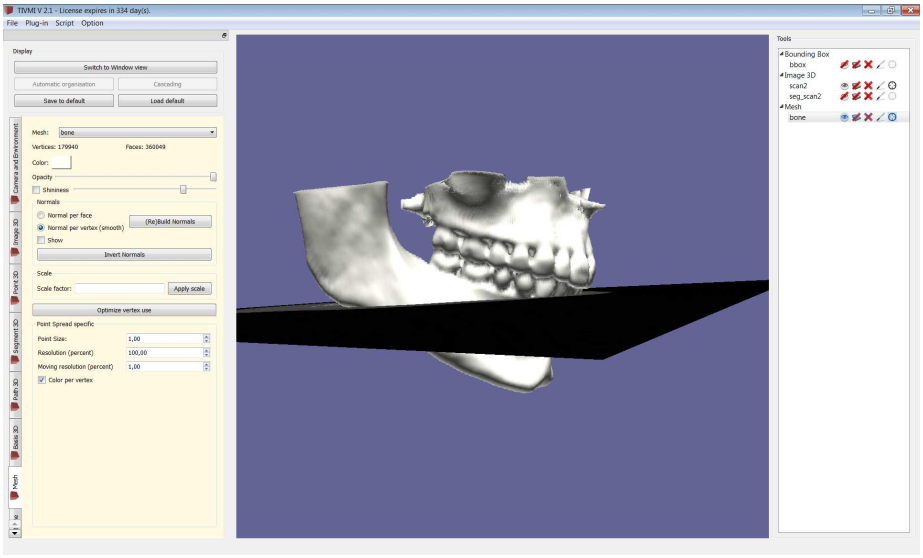

Figure 71: Create a surface from a segmented 3D image.

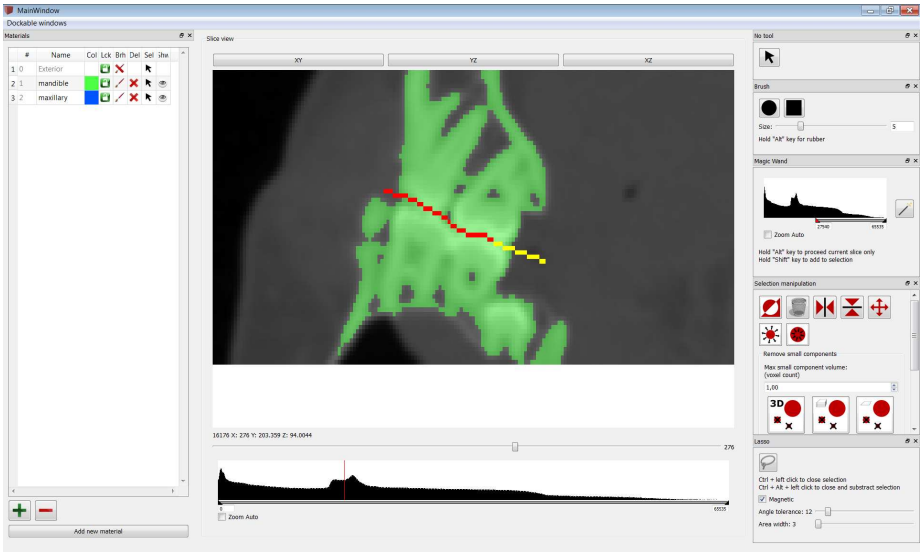

Figure 72: Magnetic lasso usage for a non trivial interface.

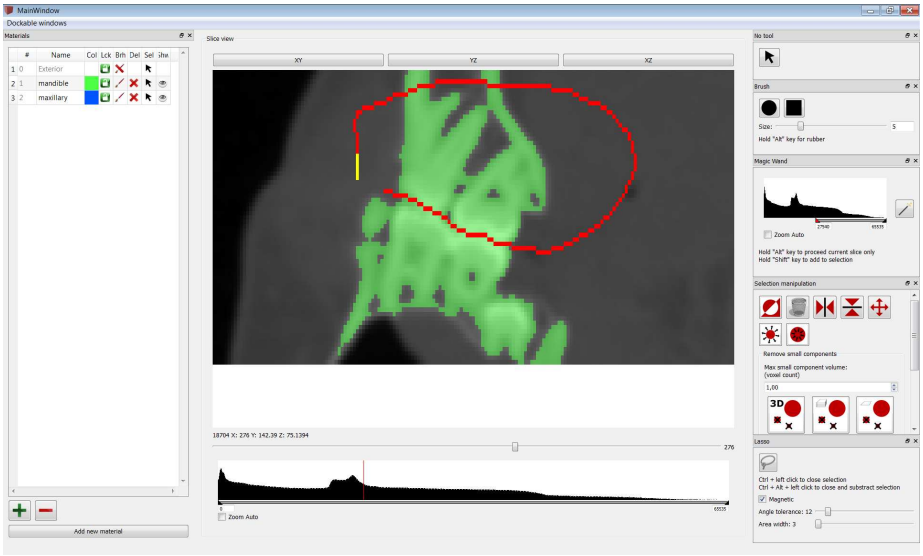

Figure 73: Magnetic lasso usage: hold mouse left button to use it as a classical lasso.

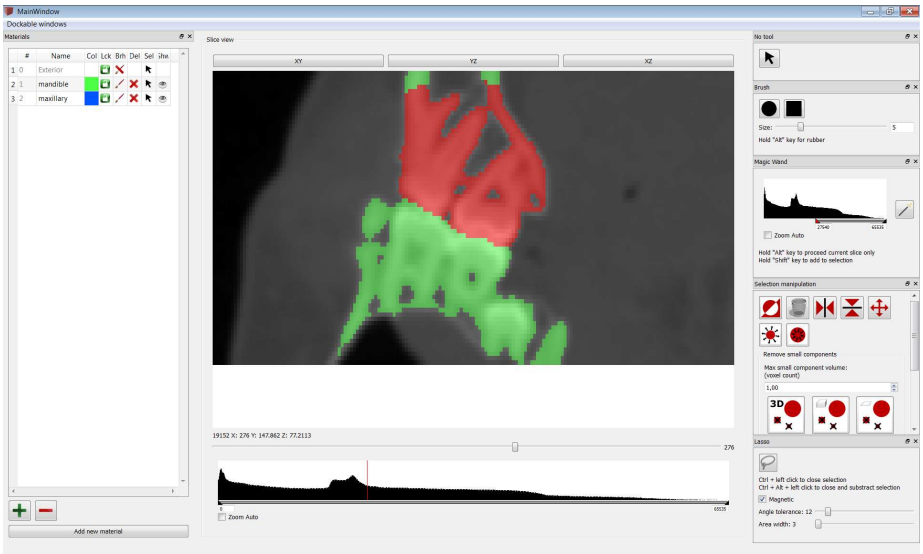

Figure 74: Magnetic lasso usage: 'crtl" click to close the magnetic lasso.

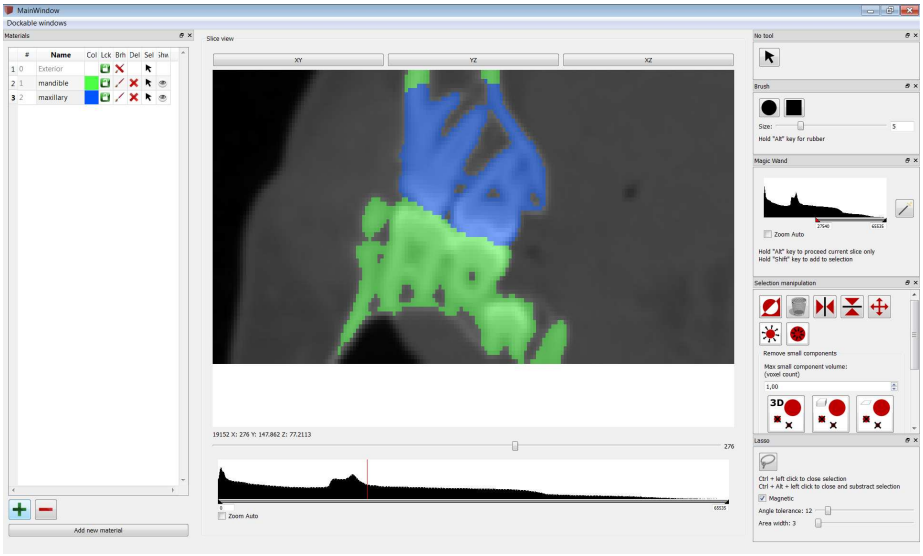

Figure 75: Add the selection to the maxillary material.

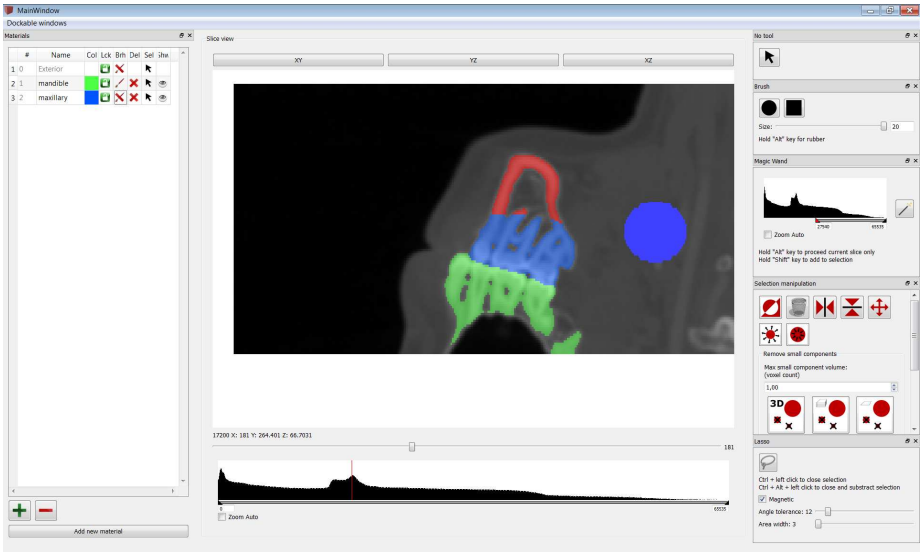

Figure 76: Use the brush to draw on remaining maxillary parts.

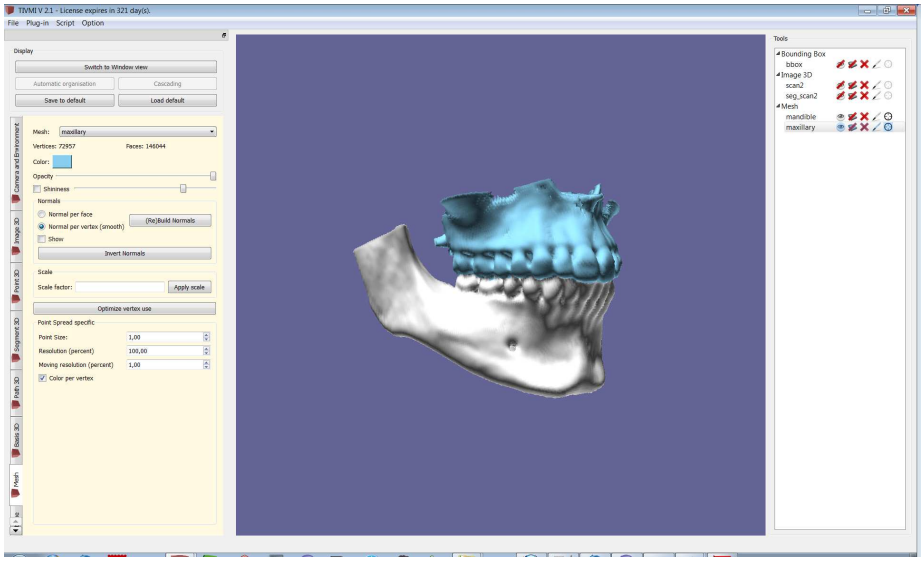

Figure 77: 3D Meshes of a mandible and a maxillary set by the segmentation editor.